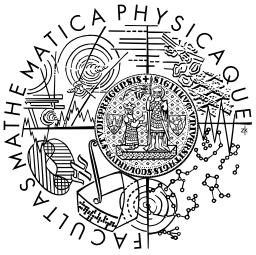


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Tadeáš Tomiška

Vývoj mobilní aplikace a generátoru otázek pro hru Desítka

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: RNDr. David Mareček, Ph.D.

Studijní program: Informatika

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád bych poděkoval vedoucímu své bakalářské práce RNDr. Davidu Marečkovi, Ph.D. za jeho trpělivost, pomoc a čas, který mi při vypracování práce věnoval. Děkuji také své rodině za všeobecnou podporu při studiu.

Název práce: Vývoj mobilní aplikace a generátoru otázek pro hru Desítka

Autor: Tadeáš Tomiška

Department: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: RNDr. David Mareček, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Tato bakalářská práce se zaměřuje na vytvoření mobilní aplikace pro Android, která umožní hrát online verzi hry Desítka od společnosti Mindok. Součástí práce je i vytvoření otázek pro hru. Ty jsou generovány pomocí dat získaných z wikipedie. Aplikace je napsána v jazyce Java a je určena pro Android verze 10 a vyše. Pro komunikaci mezi zařízeními je použita client-server architektura. Mobilní zařízení spolu komunikují prostřednictvím internetu. Aplikace má stejná pravidla jako hra Desítka a podporuje 2 herní režimy. Lze hrát v režimu online s ostatními hráči nebo v režimu friend s přáteli.

Klíčová slova: desítka generátor otázek wikipedia

Title: Development of a mobile application and question generator for the game Smart10.

Author: Tadeáš Tomiška

Department: Institute of Formal and Applied Linguistics

Supervisor: RNDr. David Mareček, Ph.D., Institute of Formal and Applied Linguistics

Abstract:

This bachelor's thesis focuses on creating a mobile application for Android that allows playing an online version of the game "Smart10" by Mindok company. The thesis also includes creating questions for the game, which are generated using data obtained from Wikipedia. The application is written in Java and is intended for Android version 10 and above. Client-server architecture is used for communication between devices. Mobile devices communicate with each other via the internet. The application follows the same rules as the game "Smart10" and supports 2 game modes. It can be played in online mode with other players or in friend mode with friends.

Keywords: smart10 question generator wikipedia

Obsah

Úvod

Desítka je společenská vědomostní hra. V České republice byla vydána v roce 2018 vydavatelstvím Mindok. (10)

Tato bakalářské práce se zaměřuje na implementaci hratelné verze hry pro mobilní zařízení s operačním systémem Android verze 10 a vyše. Součástí práce je i vytvoření dostatečného množství otázek ke hře, aby ji hráči mohli hrát opakovaně. Momentálně neexistuje žádná mobilní verze této hry.

Hra je navržena tak, aby byla co nejvíce podobná původní deskové hře. Aplikace má dva režimy hraní. Lze hrát v režimu online s ostatními hráči nebo v režimu friend s přáteli. Mobilní zařízení spolu komunikují prostřednictvím internetu. Architektura aplikace je postavena na klient-server modelu. Jako bonus byla vytvořena i offline verze hry hratelná na jednom zařízení.

1. Popis hry

Originální hra je určena pro dva až osm hráčů. Obsahuje 200 tématických okruhů, přičemž každý okruh má 10 otázek. (11) V naší implementaci je režim online určen pro 2 hráče. V režimu friend může hrát 2-5 hráčů.

1.1 Průběh hry

Hra se skládá z jednotlivých kol. V každém kole se náhodně vybere jeden okruh a hráči postupně odpovídají na otázky. Pokud hráč odpoví správně, získává bod a zůstává pro dané kolo ve hře. Odpoví-li hráč špatně, vypadává pro dané kolo ze hry a ztrácí všechny body v kole získané. Hráč se také může rozhodnout pasovat. V tom případě vypadává pro dané kolo ze hry, ale všechny body v kole získané mu zůstávají. Vítězí hráč, který jako první získá 20 bodů.

1.2 Otázky

Okruhy ve hře Desítka pokrývají široké spektrum oblastí, jako jsou například věda, historie, sport či kultura. V každém okruhu se vyskytují lehčí i těžší otázky. Zároveň se otázky snaží být zajímavé pro hráče různého věku s různými zájmy. Díky rozmanitosti otázek je Desítka nejen zábavnou hrou, ale i skvělým prostředkem pro rozvoj všeobecných znalostí a širokého rozhledu.

Obrázek 1.1: Ukázková otázka

1.2.1 Typy otázek

Otázky ve hře můžeme rozdělit na 3 typy:

- **Doplňovací otázky**
  - Jaké je hlavní město uvedeného státu?
  - Kdo je autorem tohoto díla?
- **Rozhodovací otázky** - nabízejí hráčům 2 možné odpovědi na otázku
  - Je uvedený stát členem EU?
  - Je autorem písně Pokáč nebo XindlX?
- **Řadící otázky** - vyžadují uspořádání prvků či událostí v určitém pořadí
  - Jaké je pořadí uvedených herců od nejvyššího po nejmenšího?
  - V jakém chronologickém pořadí probíhaly tyto události druhé světové války?

2. Generování otázek

Tato část se zaměřuje na tvorbu otázek do naší hry. Generování otázek je klíčovou částí vytváření herního obsahu a má významný vliv na zábavu a hratelnost hry. V našem případě bylo cílem vytvořit dostatečné množství okruhů, aby hráči mohli hrát hru opakovaně. Dalším cílem bylo udržet otázky v okruzích přiměřeně složité a pro každý okruh zajistit, že se v něm vyskytne lehčí i těžší otázka.

2.1 Jak generovat otázky?

Podívejme se na možnosti generování otázek, nad kterými jsme uvažovali.

2.1.1 Umělá inteligence

První možností, která v roce 2024 asi každého napadne je použít ChatGPT. Stačilo by vzít existující okruhy ze hry Desítka a říct ChatGPT, aby vygeneroval stejné okruhy s jinými otázkami či podobné okruhy a otázky k nim. Tento způsob s sebou však nese pár úskalí. Za prvé bychom museli vlastnoručně ověřit správnost vygenerovaných otázek. Za druhé vytvořit program, který by komunikoval s ChatGPT, procházel vygenerované okruhy s otázkami a upřesňoval požadavky na jednotlivé okruhy, je pravděpodobně těžší úkol než samotné generování otázek. Mohli bychom si natrénovat vlastní model. V tomto případě je ale problém v tom, že nemáme žádná data, na kterých by se náš model mohl učit.

2.1.2 Internetové kvízy

Další možností, nad kterou jsme uvažovali, bylo použití internetových kvízů. Na internetu najdeme celou řadu kvízů, z nichž bychom mohli vytvořit otázky. U tohoto způsobu jsou však 2 problémy. Tím prvním je, že extrakce dat z kvízů není úplně triviální, neboť jednotlivé webové stránky, na kterých se kvízy nacházejí mají různou strukturu. Výraznějším problémem je potom nekonzistence odpovědí. Například v jednom kvízu je správnou odpovědí Verne a v dalším Jules Verne. To nás vedlo k závěru, že potřebujeme nějak docílit konzistence dat.

2.1.3 Webová encyklopedie

Webová encyklopedie splňuje náš požadavek na konzistenci. Zároveň obsahuje dostatek dat pro vytvoření nemalého množství otázek. Encyklopedie navíc z převážné většiny obsahuje aktuální a ověřené informace. Celkově lze konstatovat, že webová encyklopedie představuje efektivní zdroj dat pro tvorbu otázek.

2.2 Zdroj dat

Existuje více webových encyklopedií, které jsme mohli použít. V úvahu připadaly wikipedie, wikidata či britannica. Nejvhodnější volbou pro naše potřeby se ukázala být wikipedie, a to díky následující vlastnosti:

- Wikipedia API - Tato technologie poskytuje přístup ke článkům na wikipedii. Jedná se o snadný a automatizovatelný způsob, jak získat tisíce článků, z nichž lze extrahovat data potřebná ke generování otázek. (8)

Pokud bychom použili celou wikipedii ke generování otázek, museli bychom zpracovat obrovské množství stránek. Reálně bychom potom použili pouze zlomek získaných dat. Rozhodli jsme proto najít způsob, jak snížit množství zpracovávaných stránek.

2.3 První Metoda - nejnavštěvovanější stránky

Zkusme použít nejnavštěvovanější stránky na wikipedii. Tato metoda určitě vede ke snížení počtu zpracovávaných stránek. Zároveň můžeme předpokládat, že vygenerované otázky jsou z témat, která jsou populární mezi lidmi.

2.3.1 Získání stránek

Pro nalezení nejnavštěvovanějších stránek na české wikipedii lze použít standardizované API poskytované společností Wikimedia Foundation, konkrétně následující **endpoint**.

Tento endpoint poskytuje seznam přibližně 1000 nejnavštěvovanějších článků na české wikipedii pro daný den. Pro získání dat z konkrétního dne stačí do URL adresy endpointu přidat datum ve formátu YYYY/MM/DD, například dnu 2024/01/01 odpovídá toto **URL**.

Tento přístup umožňuje rychle a efektivně získat články obsahující relevantní data pro generování otázek.

2.3.2 Parsování stránek

Získání relevantních informací přímo z textu je poměrně obtížné. Můžeme se snažit najít v textu klíčové fráze, které se opakují, například hlavní město. Museli bychom však procházet velké množství textu a pečlivě analyzovat každý odstavec, což může být časově náročné. Relevantní informace lze získat i jednodušší cestou. Stačí si všimnout, že mnoho článků na wikipedii obsahuje informační tabulku (Infobox), kde se nachází nejdůležitější informace o článku.

2.3.3 Analýza metody

Uvažme, že bychom chtěli parsovat následující Infobox. Informace vlevo představují klíče, informace vpravo představují hodnoty. Můžeme si všimnout několika problémů:

- Některé klíče mají více hodnot: Umístění - Louvre, Paříž
- Jiné klíče nemají pevně danou hodnotu: Rok vzniku - 1503-1506 (1517)
- Nějaké klíče (například rozměry) nejsou úplně vhodné pro vytvoření otázky

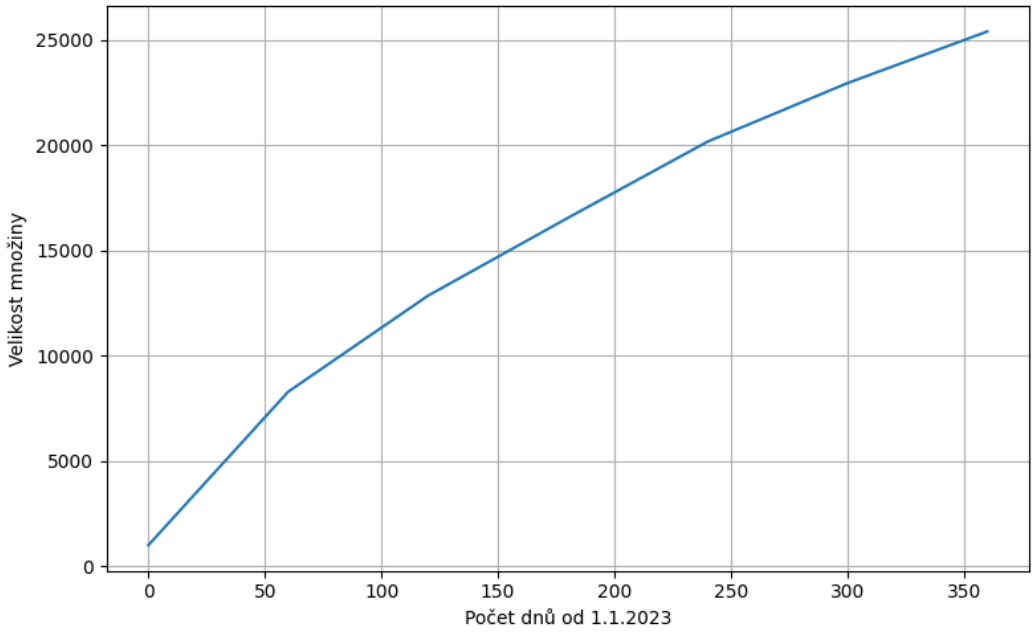
Obrázek 2.1: Ukázka infoboxu

Výše zmíněné problémy však nejsou úplně nepřekonatelné. Uvažme, že zavedeme jednoduché omezení. Naparsujeme pouze takové dvojice, kde hodnota odkazuje na právě jeden článek na wikipedii. Díky tomuto omezení navíc získáme hodnoty, které jsou obecně známé, neboť mají vlastní stránku na wikipedii. Přesto má tato metoda generování otázek problémy:

1. Nekoherence otázek - Při této metodě vytváření otázek se může stát, že vytvoříme okruh s otázkami, ve kterém se jedna z otázek bude ptát na autora Mony Lisy a druhá na autora Ústavy České republiky. To je kvůli tomu, že v infoboxech obou článků se vyskytuje klíč autor.

    - Tento problém lze odstranit tak, že vytvoříme okruh s otázkami pouze pro články, které spolu nějak logicky souvisí. U každého článku na wikipedii je v dolní části stránky uvedeno, do kterých kategorií patří. Můžeme tedy říct, že články spolu logicky souvisí, pokud mají nějakou kategorii společnou.

2. Nedostatek dat - Použít 1000 nejnavštěvovanějších článků za den zní jako dobrý nápad. Problémem však je, že nejnavštěvovanější články se den ode dne moc neliší, což dokazuje následující graf vývoje velikost množiny nejnavštěvovanějších článků za rok 2023.



Obrázek 2.2: Velikosti množíny nejnavštěvovanějších článků

2.4 Druhá metoda - kategorizace wikipedie

Jedním z problémů předchozí metody byla nekoherence otázek. Musíme nějak zajistit, aby spolu otázky v okruhu logicky souvisela. K tomu využijeme kategorizaci wikipedie, konkrétně následující **URL**. Jedná se o rodičovské URL všech kategorií na wikipedie, které představují dělení podle něčeho. Pojďme se nyní pokusit postupným zlepšováním parsovacího algoritmu vytvořit okruhy s otázkami.

2.4.1 Získání okruhů

Všimněme si, že všechny podkategorie počátečního URL obsahují slovo podle. Nabízí se tedy procházet podkategorie, dokud obsahují slovo podle. Z poslední podkategorie potom extrahovat všechny podkategorie, které představují dělení podle něčeho, to jsou naše hodnoty. Články v podkategoriích potom představují klíče. Uveďme příklad:

- Nejprve procházíme podkategorie,dokud se v nich vyskytuje slovo podle:
  Kategorie podle tvůrců/Díla podle tvůrců/Obrazy podle malíře
- Z podkategorie Obray podle malíře získáme například hodnoty:
  Obrazy Alfonse Muchy
  Obrazy Leonarda da Vinciho
- Články v podkategorii představují klíče:
  Slovanská epopej, Ilséa, princezna Tripolisská
  Mona Lisa, Dáma s hranostajem, ...

Bohužel nemůžeme získávat hodnoty pouze z posledních podkategorií, které obsahují slovo podle. Například kategorie Mistři podle sportů se zdá být vhodnou k vytvoření otázky, přestože obsahuje podkategorii Mistryně podle sportů. Na druhou stranu zkoušet vytvořit okruh z každé kategorie se slovem podle také není nejlepší nápad. Například okruh z Kategorie podle dělení by byl příliš obecný. Zvolme tedy kompromis a vytvořme okruh s otázkami z každé kategorie se slovem podle, která nezačíná názvem Kategorie.

Ještě si musíme dát pozor na to, že k některým kategoriím vede více cest. Například ke kategorii Sport podle zemí a sportů vedou následující dvě cesty: Kategorie podle místa/Kategorie podle zemí/Sport podle zemí a Kategorie podle činnosti/Kategorie podle sportů. Tento problém ale vyřešíme jednoduše. Stačí si při procházení kategorií pamatovat již navštívené kategorie.

Díky námi zvolenému způsobu výběru okruhů navíc můžeme jednoduše vytvořit text otázky. Stačí zaměnit slova před a za podle a přidat na začátek slovo určete. Například pro okruh Obrazy podle malíře lze jednoduše vytvořit otázku: **Určete malíře podle obrazu**.

2.4.2 Získání hodnot

Klíčové slovo

Zkusme získat hodnoty pro otázky z nějakého okruhu pomocí názvů podkategorií v okruhu. Zvolme například kategorii Literární díla podle spisovatelů. Nejjednoduší by bylo použít názvy všech podkategorií. Bohužel, některé názvy jako Personální bibliografie nejsou vhodné k získání hodnot. Můžeme si ale všimnout, že většina názvů má tvar: Díla jméno autora. Zároveň si všimněme, že slovo díla je i názvem okruhu. Můžeme tedy označit slovo díla za klíčové a k získání hodnot použít pouze takové podkategorie, které obsahují klíčové slovo v názvu. Pro jednoduchost neuvažujme různé tvary klíčového slova. Například slova dílo a díla jsou pro nás 2 rozdílná slova. Tato metoda není bezchybná a může nás připravit o relevatní podkategorie jako Dílo Gabriela Garcíi Marquéze či Romány Émila Zoly. Na druhou stranu pomocí této metody odfiltrujeme většinu nerelevantních kategorií. Otázkou zůstává, jak získáme klíčové slovo?

Nejpřímočařejším způsobem je extrahovat nejlevější slovo od podle. Bohužel, takto jednoduchý postup nestačí. Například pro kategorii Fiktivní postavy podle povolání s podkategoriemi fiktivní hudebníci/špióni/sportovci/atd. bychom nevybrali žádnou podkategorii. Zkusme trošku jiný přístup. Zvolme jako klíčové slovo to, které se v názvech podkategorií vyskytne nejčastěji. Tento postup funguje již poměrně dobře. Jediným problémem je, že u některých kategorií se klíčové slovo skládá z více slov. Například u Kategorie Hory a kopce podle kontinentů je klíčovým slovem Hory a kopce. Upravíme tedy náš algoritmus tak, že jako klíčové slovo uvažujeme i spojení několika slov. Zároveň chceme upřednostnit klíčové slovo složené z několika slov před jednoslovným klíčovým slovem. Každému potenciálnímu klíčovému slovu proto přiřadíme váhu rovnou počtu slov, ze kterých se skládá. Jako klíčové slovo potom určíme to, pro které bude hodnota **počet výskytů * váha** nejvyšší.

Hodnota z podstatného jména

Nyní potřebujeme získat z názvu podkategorie validní hodnotu. Ta představuje správnou odpoveď na otázku. Nabízí se použít název podkategorie bez klíčového slova. Problémem je, že takto získaná hodnota nemusí být v prvním pádu. Například pro podkategorii Obrazy Leonarda da Vinciho bychom získali hodnotu Leonarda da Vinciho. Jak získáme hodnotu v prvním pádu a jednotném čísle? Můžeme využít internetový **slovník**. Ten pro podstatná jméno v libovolném pádu vrátí jeho hodnotu v prvním pádu. Například pro kategorii Fiktivní postavy podle povolání vrátí slovník validní hodnoty detektiv/námořník/archeolog/atd. Achillovou patou slovníku je, že nezná cizí vlastní jména, tedy například jméno Leonardo da Vinci.

Nenajdeme-li hodnotu ve slovníku, máme ještě dvě možnosti. Některé podkategorie obsahují odkaz na hlavní článek. Například podkategorie Filmy Alfréda Radoka obsahuje odkaz na článek Alfréd Radok. Druhou možností je opět využít kategorizaci wikipedie. Každá podkategorie patří do nějakých kategorií. Například podkategorie Obrazy El Greca patří do kategorie El Greco, což je námi hledaná validní hodnota. Musíme pouze najít způsob, jak určit, zda název hlavního článku či některé z kategorií představuje validní hodnotu. Zvolíme následující postup. Jsou-li počáteční tři písmena každého neklíčového slova v názvu podkategorie shodná s počátečními třemi písmeny v názvu hlavního článku či některé z kategorií, máme námi hledanou kategorii. Slova, která mají méně než tři písmena, přeskočíme. Proč zrovna tři písmena? Kdybychom použili čtveřici písmen, tak již například pro podkategorii Díla Victora Huga nezískáme validní hodnotu Victor Hugo. Na druhou stranu, kdybychom použili pouze dvě písmena, tak nezískáme o moc více validních hodnot a spíše by se mohlo stát, že přiřadíme podkategorii k nevalidní hodnotě. Neobsahuje-li podkategorie validní hodnotu, vyřadíme ji.

Hodnota z přídavného jména

Jak si poradíme v případě, kdy potřebujeme získat validní hodnotu z přídavného jména? Taková situace nastane pro podkategorii Česká literární díla z okruhu Literární díla podle zemí. Slovník použít nemůžeme, neboť pro heslo česká, vrátí slovník slovo český s popiskem vztahující se k Čechám, Čechům. Pro hesla Čechám, Čechům vrátí slovník slovo Čechy. Jenže validní hodnota, kterou hledáme, je Česko nikoliv Čechy. Nabízí se opět použít kategorizaci wikipedie a procházet rodičovské kategorie, dokud nenalezneme hledanou hodnotu. Pro podkategorii Česká literární díla by prohledávání vypadalo následovně: Česká literární díla/Česká literatura/Umění v Česku/Česká kultura/Česko. Tento postup má ale jeden zásadní problém. Jak si můžeme být jisti, že Česko je skutečně námi hledaná validní hodnota? Uvažme podkategorii Skotská hudba z okruhu Hudba podle národů. Postupným procházením rodičovských kategorií dostaneme hodnotu Skotsko. Jenže Skotsko je země a my hledali národ, takže validní hodnota je Skotové. Museli bychom tedy nějak ověřovat získané hodnoty. Třeba podle toho, do kterých kategorií patří. Dále bychom museli řešit synonyma, neboť získaná hodnota může patřit do kategorie stát, ale my hledáme zemi. Celkově je tento postup poměrně složitý, proto zvolíme jinou cestu. Upravíme text otázky.

Zatím jsme uvažovali jen jeden způsob vytvoření textu otázky. Například pro okruh Literární díla podle zemí jsme počítali s otázkou: **Určete zemi podle literárního díla**. Problémem je, že podkategorie v okruhu mají tvar Italská/-Česká/Německá literární díla a my neumíme z přídavných jmen efektivně získat validní podstatná jména. Proto pro takové okruhy upravíme text otázky takovým způsobem, aby odpověď vyžadovala přídavné jméno. Text otázky bude začínat textem Jedná se o, následně do hranatých závorek dáme hodnotu, jejíž přídavné jméno budeme očekávat v odpovědi, na konec otázky přidáme text, který se nachází v názvu okruhu před slovem podle. Pro uveden příklad bude nová otázka vypadat následovně: **Jedná se o [země] literární dílo**. Jako odpověď budeme od uživatele očekávat přídavná jména české/italské/německé/atd.

Jak určíme, zda je hodnota podstatné nebo přídavné jméno? Mohli bychom opět použít internetový slovník. My ale použijeme jednodušší cestu. Pokud je hodnota před klíčovým slovem, budeme ji považovat za přídavné jméno. Je-li hodnota za klíčovým slovem, považujeme ji za jméno podstatné. Pro úplnost dodejme, že nemůžeme míchat podkategorie, kde je hodnotou podstatné/přídavné jméno. Například pro okruh Skladby podle autorů, kde má většina podkategorií tvar Skladby jméno autora, nezahrneme podkategorii Anonymní hudba.

2.4.3 Získání klíčů

Nejtěžší fází naší metody je získávání klíčů pomocí článků v podkategoriích. Pro získání korektních klíčů musíme překonat několik překážek.

Nepřítomnost článků

Do podkategorie Fiktivní detektivové podle povolání určitě patří článek o Sherlocku Holmesovi. Sherlock Holmes je však tak rozšířený pojem, že si vysloužil vlastní podkategorii. Abychom získali článek o Sherlocku Holmesovi, mohli bychom projít celou podkategorii Sherlock Holmes. Tím bychom však získali i další články jako Profesor Moriarty. Tato postava však není detektivem. Musíme proto použít jinou strategii. Místo procházení celé podkategorie Sherlock Holmes jednoduše vyzkoušíme, jestli na wikipedii neexistuje článek Sherlock Holmes. Pokud existuje, tak jej přidáme jako jeden z možných klíčů.

Pro kategorii Mistři podle sportů nastane trošku jiná situace. Uvažme, že chceme získat klíče pro kořenovou podkategorii Mistři v akrobatickém lyžování. Tato podkategorie neobsahuje žádný článek. Má pouze jednu podkategorii s názvem Mistři světa v akrobatickém lyžování. Všimněme si, že název Mistři světa v akrobatickém lyžování obsahuje všechna slova názvu Mistři v akrobatickém lyžování. Toho bychom mohli využít a získávat data i z podkategorií, které obsahují v názvu všechna slova kořenové podkategorie. Tento postup však selže pro kategorii Jeskyně podle zemí. Chtějme získat klíče pro kořenovou podkategorii Jeskyně v Evropě s podkategoriemi Jeskyně ve Španělsku/Francii/atd. Tyto podkategorie neobsahují v názvu všechna slova kořenové podkategorie. Přesto mají názvy všech podkategorií jednu věc společnou, všechny obsahují v názvu klíčové slovo jeskyně. Toho využijeme a budeme získávat data ze všech podkategorií, které obsahují v názvu klíčové slovo.

Irelevantní články

Pomocí postupu výše jsme získali pro danou podkategorii větší množství článků. Bohužel se může stát, že ne všechny získané články jsou pro daný okruh relevantní. Uvažme kategorii Pohoří podle kontinentů. Chtějme získat klíče pro podkategorii Pohoří v Severní Americe. Získáme články ze všech podkategorií s klíčovým slovem pohoří, tedy i článek Appalečské pohoří/Hory a kopce v Appalačském pohoří/Mount Mitchell. Jenže Mount Mitchell je názvem hory, nejedná se tedy o relevantní klíč. Naštěstí lze získání takovýchto irelevantních článků jednoduše předejít. Stačí se pro každou podkategorii podívat, jestli není také samostatným článkem na wikipedii. Pokud ano, tak podkategorii neprocházíme. Je totiž pravděpodobné, že podkategorie bude obsahovat články, které pro náš okruh již nejsou relevantní. Například pro podkategorii Pohoří v Severní Americe přidáme podkategorii Appalačské pohoří pouze jako jeden článek, zatímco z podkategorie Pohoří v Mexiku přidáme všechny články, protože článek Pohoří v Mexiku neexistuje.

Může se nám také stát, že podkategorie obsahuje sama od sebe irelevatní článek. Například podkategorie Jeskyně ve Španělsku pro okruh Jeskyně podle zemí obsahuje článek Národní archeologické muzeum (Madrid). Jak určíme, že tento článek je irelevantní? To není tak jednoduché. Mohli bychom prověřit kategorie, do kterých článek patří, což jsou Muzea v Madridu, Muzea založená roku 1867 a Jeskyně ve Španělsku. To, že článek patří do 2 kategorií, jejichž název začíná slovem muzea, je trošku podezřelé, ale nemusí to znamenat, že je článek irelevantní. Pro určení relevantnosti článku bychom si museli nějak specifikovat, čeho se týká. Nabízí se použít stránku článku na wikipedii určenou pro mobilní zařízení. Zaměříme se na to hned v další podsekci.

Specifikace článku

K lepšímu zařazení článku bychom mohli použít stránku wikipedie určenou pro mobilní zařízení. Její základní url má tvar cs.m.wikipedia.org. Porovnejme základní **verzi** stránky na wikipedii s **verzí** pro mobilní zařízení. Můžeme vidět, že stránka pro mobilní zařízení obsahuje pod názvem článku jeho definici. Pro specifikaci článku můžeme použít první podstatné jméno, které se v článku vyskytne a pro tvorbu otázek použít pouze takové klíče, jejichž specifikace se vyskytla vícekrát než nějaký stanovený limit. Tím předejdeme tomu, že jako klíč použijeme nějaký irelevantní článek.

Problémem tohoto postupu je, že některé články žádnou specifikaci neobsahují, tudíž bychom je museli považovat za irelevantní. Dále bychom také museli řešit synonyma, neboť například specifikace země a stát mají stejný význam. Největším problémem je, že použití specifikace u některých článků selhává. Například pro kategorii Fiktivní postavy podle povolání máme podkategorii Fiktivní špióni. Ta obsahuje klíč James Bond. Tento článek má následující specifikaci: "mediální franšíza o britském tajném agentovi, založená knihami Iana Fleminga". Prvním podstatným jménem ve specifikaci je franšíza. Jenže franšíza má pramálo společného s podstatnými jmény postava či špión. Z tohoto důvodu jsme se nakonec rozhodli klíče nespecifikovat a spolehnout se na to, že irelvantní články budou patřit mezi méně známé články, na jejichž odstranění se nyní zaměříme.

Filtrování klíčů

Wikipedie obsahuje opravdu velké množství článků. Z toho důvodu se nám klidně může stát, že získáme i téměř neznámé klíče. Tento problém vyřešíme pomocí api, jež wikipedie poskytuje. To umožňuje získat počet návštěv daného článku za určité časové období. Api zobrazuje počet návštěv den po dni nebo po měsících. My použijeme měsíční verzi. Například pomocí následujícího **url** získáme měsíční počty návštěv článku Anglie za rok 2023. Zbývá nastavit hranici počtu návštěv, při jejímž překročení můžeme klíč považovat za známý. Určit hranici bez jakýchkoliv dat je však těžké, proto použijeme více hodnot. Konkrétně vytvoříme otázky z článků, jejichž průměrná měsíční návštěvnost je postupne 128, 256, 512, 1024, 2048. Používáme data návštěvnosti za rok 2023.

Triviální klíče

Dalším problémem, který musíme vyřešit je, že klíč a hodnota mají stejný kořen slova. Například pro kategorii Stavby podle autora bychom mohli vygenerovat otázku s klíčem Eiffelova věž a hodnotou Stavby Gustava Eiffela. Tyto dvojice jsou nevhodné pro tvorbu otázky, neboť by se jednalo o velmi jednoduché otázky. Pro odstranění problému by bylo ideální zamítnout klíče, které mají stejný kořen jako nějaké neklíčové slovo hodnoty. Získání kořene slova však není úplně jednoduché, proto použijeme trošku jiný postup. Pro každé slovo v hodnotě vezmeme jeho tři počáteční písmena. Nyní pro každé slovo v klíči ověříme, že neobsahuje jako podřetězec danou trojici. Pokud klíč danou trojici písmen obsahuje, zamítneme jej. Důvod pro 3 písmena je podobný jako v případě hodnot. Určitě nemůžeme vzít méně písmen. Kdybychom použili čtveřici písmen, tak již nezamítneme klíč Asijsko-pacifický region pro hodnotu Geografie Asie. Na druhou stranu, kdybychom použili pouze 2 písmena, tak je již vysoká pravděpodobnost, že nesprávně zamítneme validní klíč.

Klíče s více hodnotami

Až do této chvíle jsme tiše předpokládali, že při zpracování nějaké kategorie má každý klíč přesně danou hodnotu. To však pro některé kategorie neplatí. Například kategorie Fiktivní postavy podle povolání obsahuje podkategorie Fiktivní archeologové a Fiktivní antropologové. Obě tyto podkategorie obsahují článek s názvem Daniel Jackson. Existuje tedy více správných odpovědí na otázku Určete povolání podle fiktivní postavy s klíčem Daniel Jackson. Naštěstí tento problém lze odstranit jednoduše zamítnutím všech klíčů, jež se vyskytnou ve více podkategoriích.

2.4.4 Tvorba otázek

Text otázky

Ve hře používáme 2 typy otázek. První typ je pro otázky, jejichž hodnotou je podstatné jméno. Druhý typ je pro otázky, jejichž hodnotou je přídavné jméno. Text otázek vytváříme pomocí názvu okruhů. Připomeňme si znovu texty obou druhů otázek.

- **Okruh:** Obrazy podle malíře
  Otázka: Určete malíře podle obrazu

- **Okruh:** Literární díla podle zemí
  Otázka: Jedná se o [země] literární dílo

Můžeme si všimnout, že pro oba typy otázek potřebujeme převést slova z názvů okruhů do jiných pádů. Konkrétně pro první typ otázek potřebujeme převést slovo malíře do čtvrtého pádu a slovo obrazy do druhého pádu. U druhého typu otázek potřebujeme převést slovo zemí do prvního pádu a slova literární díla do čtvrtého pádu, vždy čísla jednotného. Opět využijeme již zmíněný internetový **slovník**. Ten pro podstatná jména pracuje dobře. Menší překážkou je, že některá přídavné jméno převádí přímo na podstatné jméno, od kterého bylo odvozeno. Například pro slovo italská vrátí slovník slovo Itálie. Naštěstí slovník vrátí pro přídavné jméno i další slova s ním spojená. Mezi těmito slovy již můžeme najít hledané přídavné jméno.

Výběr klíčů

Každý okruh obsahuje několik hodnot s několika klíči. Pro každý klíč máme uloženou jeho specifikaci a průměrnou měsíční návštěvnost. Nejprve spočítáme počet výskytů nejčastější hodnoty. Vyskytne-li se nejčastější hodnota ve více než 50 procentech klíčů, tak otázky nevytváříme. Na většinu otázek by totiž byla stejná odpověď. klíče, je-li jich aspoň 10, seřadíme podle návštěvnosti. Otázky vytvoříme následujícím způsobem. Určíme velikost kroku k jako počet klíčů děleno 10. N-tá otázka potom obsahuje klíče na indexech n, n+k, n+2k, atd. Tím docílíme toho, že v každý okruh bude obsahovat více i méně známé klíče.

Přidání možností

Originální hra má výhodu v tom, že určení správnosti odpovědi závisí přímo na hráčích. Ptá-li se hra na autora obrazu a jako správná odpoveď je uveden Leonardo da Vinci, hráč si uzná odpověď, i když řekne pouze da Vinci. V našem případě musíme uznání správnosti odpovědi zakódovat do programu. Zatím jsme předpokládali, že budeme chtít po uživateli, aby napsal správnou odpověď. To je ale poměrně nepraktické. Jak bychom si poradili v případě, kdy jako odpověď očekáváme Česko a uživatel napíše Česká republika. Další překážkou je, že tento přístup je pro uživatele nekomfortní. Jen si představme, že máme jako odpověď na otázku zadat nějaké složité cizí vlastní jméno. Proto jsme se rozhodli poskytnout ke každé otázce 4 možné odpovědi. Tři špatné odpovědi vybíráme náhodně ze všech ostatních hodnot. To s sebou nese jedno omezení. Můžeme vytvářet pouze otázky z okruhů, které obsahují alespoň 4 hodnoty.

2.4.5 Shrnutí procesu

Pojďme si jednoduše shrnout celý proces generování otázek.

1. Pomocí kategorií na wikipedii získáme otázkové okruhy.

2. Pro každý okruh získáme z jeho podkategorií hodnoty. V případě, že je hodnotou podstatné jméno, které nenajdeme ve slovníku a které nemá vlastní kategorii či článek na wikipedii, tak hodnotu neuvažujeme.

3. Pro daný okruh získáme z každé jeho podkategorie klíče. Zamítneme nevalidní klíče a klíče s menší průměrnou měsíční návštěvností než současná hranice.

4. Vytvoříme text otázky a vybereme klíče k otázce. Ke každému klíči v otázce také náhodně vybereme tři špatné hodnoty.

2.4.6 Finální úpravy

Při samotné implementaci generování otázek jsme narazili ještě na pár drobných problémů, které jsme museli odstranit.

Zrychlení generování

Prvním problémem se ukázalo, že pokud při generování otázek použijeme jedno jediné vlákno, tak je proces velmi pomalý. Z tohoto důvodu jsme se rozhodli pro paralelizaci. Použijeme více vláken, které budou paralelně zpracovávat okruhy a vytvářet otázky.

Získání správných tvarů slov

Další problém tkví v následujícím. I když zmíněný slovník poskytuje tabulku se správným skloňováním podstatných a přídavných jmen, napsat program, který k této tabulce přistupuje, je složitější. Samotné získání tabulky není možné prostou metodou GET, ale vyžaduje interakci se serverem. Například v Javě bychom mohli vytvořit klienta, který otevře webový prohlížeč (například Google Chrome) pomocí Selenium a následně simuluje stisk tlačítka, které zobrazí tabulku. Takový postup ale vyžaduje pevné zadrátování lokální adresy webového prohlížeče přímo do kódu, což omezuje transparentnost programu. Z toho důvodu jsme se rozhodli pro získání tabulek skloňování podstatných a přídavných jmen použít **wikislovník**. Z něj můžeme tabulku skloňování získat jednoduchým GET requestem.

Dalším problémem bylo, že jsme občas vygenerovali nesprávný text otázky. Například pro podkategorii Přístavy podle řek jsme vygenerovali otázku: "Poznejte Řeka podle přístavu". To bylo z toho důvodu, že internetový slovník nám pro slovo řek vrátí 2 hesla: Řek, řeka. My jsme se podívali na všechny tvary prvního vráceného hesla, a pokud obsahovalo variantu řek (case insentive), tak jsme vrácené heslo použili v naší otázce. Po vyzkoušení našeho generátoru na malém vzorku kategorií jsme si všimli, že většina názvů kategorií obsahuje maximálně 5 slov. V případě, že správnou odpovědí na otázku je přídavné jméno, je většina názvů kategorií čyřslovná (za slovem podle je pouze podstatné jméno). Nachází-li se před či za slovem "podle"2 slova, jedná se velmi často o spojení přídavného a podstatného jména. Tohoto faktu jsme se rozhodli využít a vytváříme otázky pouze pro okruhy, jejichž název obsahuje maximálně 5 slov (4 slova pro adjektivní typ otázek) a v nichž se dvouslovná spojení skládají z přídavného a podstatného jména. Díky tomu můžeme využít toho, že slova nalevo od podle jsou vždy v prvním pádu a slova napravo od podle jsou vždy ve druhém pádu. Můžeme tedy

v tabulce skloňování porovnat přímo tvar pro daný pád. Tím vyřešíme problém s řekou.

Posledním problémem bylo získání správných tvarů hodnot. U hodnot totiž nevíme, ve kterém jsou pádu. Uvažme kategorii Pohoří podle zemí s podkategorií Pohoří ve Francii. Slovník nám pro slovo Francii vrátí hesla francium a Francie. Oba klíče obsahují tvar francii/Francii, a protože první vracené slovo je francium a my porovnáváme case insensitive, budeme jako odpověď na otázku očekávat hodnotu francium. Tento problém vyřešíme tak, že upřednostníme hesla s velkými písmeny. Ještě zbývá odpovědět na otázku, proč při porovnávání nerozlišujeme velká a malá písmena? To je kvůli tomu, že název každé kategorie na wikipedii začíná velkým písmenem. Museli bychom tudíž hledání správného tvaru slova pro počáteční slovo kategorie ošetřit individuálně, čemuž jsme se použitím case insensitive přístupu vyhnuli.

Pro úplnost dodejdeme, že internetový slovník pro slovo "let"vrátí pouze hesla let, letový. Budeme-li mít kategorii s názvem Bitvy podle let, nebudeme schopni pro ni vytvořit otázku. Tomu jsme předešli tak, že jsme do programu uložili, že čtvrtý pád pro slovo let je rok.

Vygenerování otázek

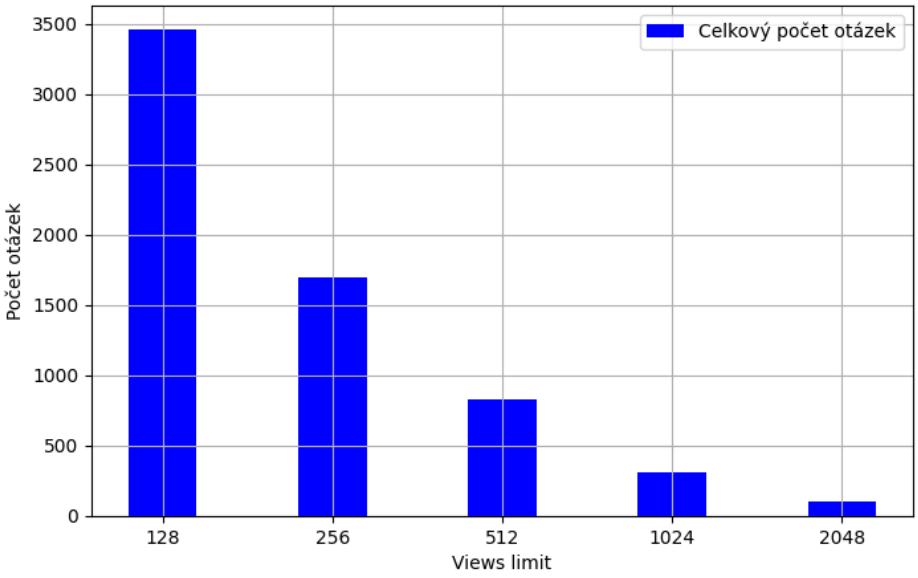
Vygenerování okruhů z rodičovské kategorie "Kategorie podle dělení"se nám povedlo až na třetí pokus. Nejprve jsme nastavili operační paměť programu na 8GB, později na 16GB. Program v obou případech selhal kvůli nedostatku místa na heapu. Řešením by bylo generovat otázky pro různé kategorie postupně, my jsme se však rozhodli využít fakultní server. Na něm již nás program za 66 minut doběhl. V následující sekci si zanalyzujeme otázky, které jsme vygenerovali.

2.5 Analýza otázek

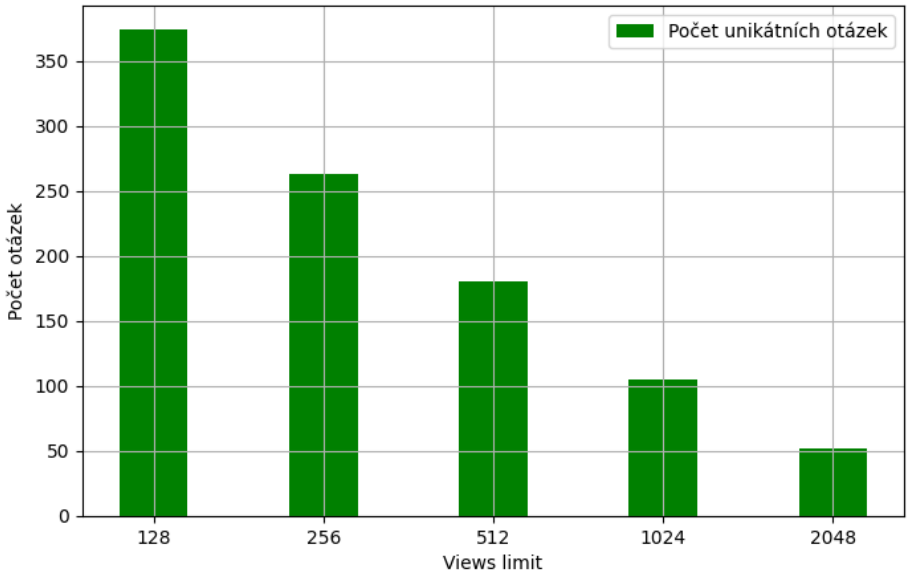
V této části se podíváme na otázky, jež jsme vygenerovali využitím kategorizace wikipedie. Otázky jsme získali ze všech podkategorií následujícího **url**. Vygenerované otázky jsou dostupné **zde**.

2.5.1 Získané otázky

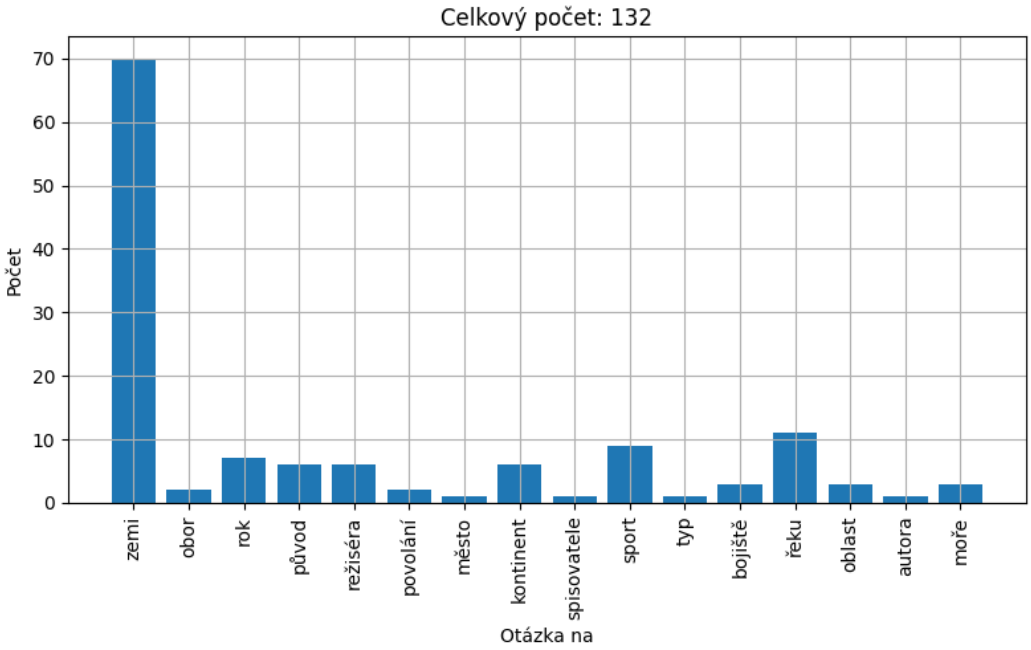
Následující grafy přiblužují vygenrevané otázky. V souboru **output.log** je poté zaznamenám celý průběh generování. Výsledné grafy nejsou nikterak překvapující. Na prvních dvou grafech si můžeme všimnout, že s rostoucí hranicí minimálního limitu měsíčních zobrazení článku počet vytvořených otázek klesá. To samé platí i pro unikátní počet otázek. Otázkou je, které vygenerované otázky použít ve hře. Originální hra obsahuje 200 otázek. Proto jsme se rozhodli použít ve hře otázky s hranicí počtu zhlédnutí 1024. Jedná se zhruba o 300 otázek, s přibližně 100 unikátními otázkami. Zbylé 2 grafy blíže přibližují tyto otázky. Můžeme si všimnout, že velká část otázek požaduje správné přiřazení země. Na druhou stranu najdeme i další otázky. Například otázky: Poznejte rok podle bitvy, Určete českého režiséra podle filmu či Doplňte řeku podle sídla.



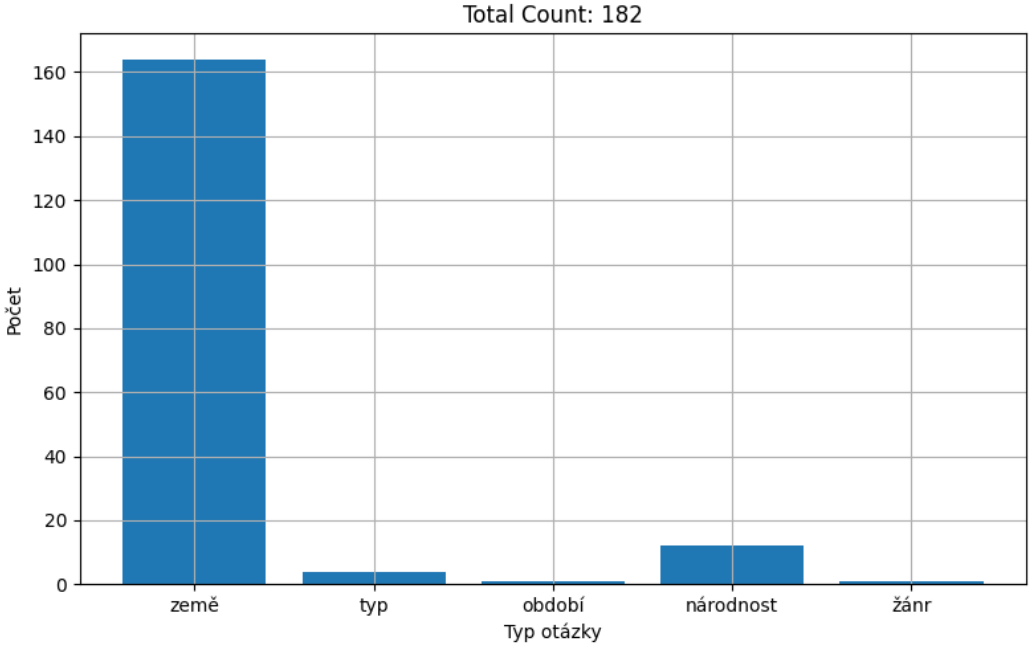
Obrázek 2.3: Celkový počet otázek



Obrázek 2.4: Počet unikátních otázek



Obrázek 2.5: Otázky na podstatné jméno



Obrázek 2.6: Otázky na přídavné jméno

2.5.2 Korektnost otázek

Korektnost otázek můžeme určit z několika úhlů pohledů. Prvním měřítkem je, jestli jsou texty otázek a nabízené odpovědi na otázku gramaticky správně. V tomto ohledu otázky prošly. Dalším měřítkem je logická správnost nabízených odpovědí na otázku. Tím je myšleno, že pokud otázka vyžaduje doplnění země, budou mezi nabízenými hodnotami skutečně země. V tomto ohledu většina otázek také prošla. U některých otázek jsou mírné nuance. Například otázka číslo 206: "Jde o [země] kuchyni?". Hned první klíč je Carbonara a nabízenými odpověďmi na otázku jsou: židovskou, italskou, středomořskou, balkánskou. Některé nabízené odpovědi nejsou skutečnými státy, přesto nejsou úplně zcestné. U této otázky bychom mohli řešit ještě jeden problém spočívající v zavádějících odpovědích. Správnou odpovědí je "italskou", ale odpověď "středomořskou"by také nemusela být nutně považována za chybnou. Musím ale konstatovat, že i v tomto ohledu chybných otázek je velmi malé množství. Posledním měřítkem, podle kterého můžeme otázky porovnat je správná úroveň obtížnosti. V tomto ohledu si otázky opět vedou poměrně dobře. Osobně jsem v některých okruzích, zaznamenal 10 správných odpovědí. Naopak v okruzích, které jsou mimo spektrum mých zájmů jsem měl pouze 2 až 3 správné odpovědi.

2.5.3 Formát uložení otázek

Ještě jsme se nijak nezabývali otázkou, jak jsme vlastně vygenerované otázky uložili. Rozhodli jsme se použít JSON. Jedná se o obecně rozšířený formát určený pro přenos dat. Následující obrázek obsahuje úvodní část první použité otázky.


```
{
  "text": "Určete zemi podle historického území",
  "questions": [
    {
      "key": "Sudety",
      "values": [
        "Francie",
        "Chorvatsko",
        "Česko",
        "Rusko"
      ],
      "correctIndex": 2
    },
```


Obrázek 2.7: Ukázka uložení otázky

2.5.4 Pouze jeden typ otázek

V úvodní kapitole jsme si rozdělili otázky ve hře desítka na 3 typy. My jsme vygenerovali pouze doplňovací otázky. Pojďme se podívat, jak bychom mohli vygenerovat zbylé 2 typy otázek a zdůvodnit si, proč jsme tak neučinili.

Rozhodovací otázky

Tento typ otázek nabízí hráčům 2 možné odpovědi na otázku. Nejčastěji se jedná o odpoveď ANO/NE. Otázky tohoto typu bychom mohli ze získaných dat poměrně jednoduše vygenerovat. V každém okruhu se vyskytují alespoň 4 hodnoty a pro každou hodnotu máme několik klíčů. Je-li nějaká hodnota asociována s alepoň třemi klíči, mohli bychom vytvořit otázku: Jedná se o [klíč] [hodnoty]? Uveďme příklad, máme okruh Obrazy podle malířů s hodnotou Leonardo da Vinci. Pro tuto hodnotu máme několik klíčů. Můžeme tedy vytvořit otázku: Jedná se o obraz Leonarda da Vinciho? a jako klíče vybrat da Vinciho obrazy i obrazy jiných malířů. Na druhou stranu téměř všechny klíče z tohoto okruhu (počet klíčů modulo 10 není použit) jsou použity u doplňovacího typu otázek. My bychom tedy pouze zrecyklovali již získaná data, což nám přišlo nevhodné.

Řadící otázky

Tento typ otázek vyžaduje uspořádání prvků či událostí v určitém pořadí. Například seřadit hory podle výšky nebo státy podle rozlohy. K získání dat pro tyto otázky bychom mohli použít seznamy na wikipedii. Například Seznam nejvyšších hor pro seřazení hor podle výšky. Problémem je, že tento seznam obsahuje pouze 50 nejvyšších hor a rozdíl mezi nejvyšší a nejnižší horou na seznamu je menší než 1500. Takto vytvořená otázka by tedy byla opravdu těžká. Jediné užitečné seznamy, které jsme našli, jsou v kategorii Seznamy států světa. Rozhodli jsme se ale, že nechceme vytvořit typ otázky, v němž bychom po hráčích vždy chtěli seřadit státy podle nějaké charakteristiky. Důvodem k tomuto rozhodnutí bylo i to, že otázek týkajících se států máme již poměrně velké množství.

2.5.5 Uživatelská dokumentace

Aplikace je napsaná v základní verzi Javy 8 a používá nástroj Maven pro správu závislostí a sestavení projektu. V případě, že si uživatel nechce instalovat maven, může použít i jar soubor s názvem generator-jar-with-dependencies.jar umístěný přímo v adresáři QuestionGenerator. Aplikace přijímá následující parametry z příkazové řádky:

- **–startingTitle <title>**
  - Název počáteční kategorie na Wikipedii
  - Výchozí hodnota je Kategorie:Kategorie_podle_tvůrců
- **–categoriesDirectory <dir>**
  - Cesta k adresáři, kde se uloží získané kategorie
  - Výchozí hodnota je output/categories

- **–questionsDirectory <dir>**
  - Cesta k adresáři, kde se uloží vygenerované otázky
  - Výchozí hodnota je output/questions
- **–categoriesViewsLimit <limit>**
  - Hranice pro považování článku za relevantní
  - Výchozí hodnota je 128
- **–questionsViewsLimit <limit1>,<limit2>,...**
  - Hranice pro přidání článku do otázky
  - Výchozí hodnoty jsou 128, 256, 512, 1024, 2048

Jar

JAR (Java Archive) je archivační formát, který se používá k seskupení souborů a balíčků do jediného archivu. Často se používá pro distribuci a nasazení Java aplikací. Pro spuštění stačí mít naistalovanou Javu verze 8 a výše. Základní příkaz spuštění je následující, stačí zadat v adresáři QuestionGenerator. V případě, že chceme program spustit s upravenými parametry, stačí je přidat na konec příkazu.

```
java -jar generator-jar-with-dependencies.jar
```

```
java -jar generator-jar-with-dependencies.jar --startingTitle ...
```

Maven

Apache Maven je nástroj pro správu projektů a automatizaci procesu vývoje softwaru v jazyce Java. Jeho hlavním účelem je spravovat závislosti projektu, kompilovat zdrojové kódy, testovat a balit výsledný software do distribuovatelné formy. Maven je založen na konceptu projektového souboru (pom.xml), který obsahuje konfiguraci projektu a informace o závislostech. Pomocí tohoto souboru Maven identifikuje potřebné knihovny a spravuje jejich stahování a aktualizace z repozitářů. Abychom mohli pracovat s nástrojem Maven, musíme si jej nejdřív nainstalovat. To lze na Linuxu nebo na WSL (Windows Subsystem for Linux) s distribucí Ubuntu pomocí následujících příkazu:

```
sudo apt install maven
```

Nyní již můžeme s nástrojem Maven pracovat. Můžeme použít například příkaz test, který spustí 3 ukázkové testy. První test vygeneruje otázky pro kategorii Fiktivní postavy podle povolání. Druhý test vygeneruje otázky pro kategorii Romány podle zemí. V tomto případě bude správnou odpovědí přídavné jméno. Poslední test je komplexnější. Ten vygeneruje otázky pro všechny podkategorie v Kategorii podle tvůrců.

```
mvn test
```

Pro spuštění hlavní metody stačí zadat následující příkazy.

```
mvn compile
mvn exec:java
```

Pro vytvoření nového souboru jar můžeme použít následující příkaz. Nově vytvořený soubor bude v adresáři target.

```
mvn clean compile assembly:single -DskipTests
```

2.5.6 Programátorská dokumentace

Samotný program se skládá z několika tříd v několika balíčcích. Podrobná dokumentace je v adresáři javadoc na následujícím **odkazu**.

desitka.dictionaries

Tento balíček slouží k získání daného pádu pro zadané slovo. Obsahuje 2 třídy, které spolu navzájem spolupracují.

- **Dictionary.java**
  - Tato třída má na starost komunikaci s internetovým **slovníkem**. Jejím účelem je vrátit pro zadané slovo jeho hodnotu v prvním pádu.
- **Wiktionary.java**
  - Tato třída má za úkol vrátit pro dané slovo jeho variantu v požadovaném pádu. K tomu používá **wikislovník**.

desitka.question

Tento balíček má na starost vytvoření textu otázky, vytvoření samotné otázky z klíčů a hodnot. Balíček také obsahuje třídu, která vygenerované otázky přesune do cílového adresáře.

- **Question.java**
  - Tato třída reprezentuje otázku ve formátu JSON. K jejímu uložení používáme knihovnu Gson, což je knihovna pro zpracování JSON dat v Javě.
- **QuestionCollector.java**
  - Tato třída posbírá vygenerevané otázky ve všech kategoriích a přemístí je do cílové destinace.
- **QuestionCreator.java**
  - Tato třída má za úkol z klíčů a hodnot vytvořit otázky.
- **QuestionTextCreator.java**
  - Úkolem této třídy je vytvořit text otázky. Jediné, co k tomu třída potřebuje je název kategorie a informace, jestli má být odpovědí na otázku podstatné či přídavné jméno.

desitka.wiki

Tento balíček má na starost získání klíčů a hodnot z wikipedie. Obsahuje 1 třídu a 2 další balíčky.

- **PageViewer.java**
  - Tato stránka vrací pro zadaný článek průměrný měsíční počet návštěv za rok 2023.
- **desitka.wiki.getters**
  - Tento balíček je hlavní částí programu. Skládá se ze čtyř tříd, kterí mají společně na starost získání klíčů a hodnot z wikipedie.
  - **CategoriesWikiGetter**
    - Tato třída má za úkol projít wikipedie a získat z ní všechny kategorie, které obsahují slovo podle
  - **ValuesWikiGetter**
    - Tato třída má za úkol získat ze vhdných kategorií hodnoty do otázky. Tato třída také volá QuestionTextCreator, čímž zajišťuje vytvoření textu otázky.
  - **KeysWikiGetter**
    - Tato třída má na starost získat z podkategorií s hodnotou vhodné klíče.
  - **WikiGetter**
    - Toto je abstraktní třída, od které dědí všechny 3 předchozí třídy. Třída má na starost komunikaci s Wikipedií. Využívá 3 následující endpointy pro získání podkategorií, rodičovských kategorií a hlavního článku. Ukažme si je na příkladu. Uvažme třeba následující kategorii: Kategorie_Obrazy_Leonarda_da_Vinciho
    - **Podkategorie**.
    - **Rodičovské kategorie**.
    - **Hlavní článek**.
- **desitka.wiki.types**
  - Tento balíček obsahuje třídy, které reprezentují různé objekty na wikipedii.

desitka

Přímo v balíčku desitka máme ještě 3 třídy. Dvě jsou spíše pomocné a třetí představuje takový wrapper, jenž zapouzdřuje celou logiku generování otázek.

- **ArgumentParser.java**
  - Tato třída slouží k naparsování parametrů při spuštění programu z příkazové řádky.

- **JsoupGetter.java**
  - Toto je komunikační třída. Jejím úkolem je získat ze zadaného URL hodnotu v Json formátu. K tomu využívá knihovnou Jsoup.
- **QuestionGenerator.java**
  - Tato třída postupně volá metody již zmíněných tříd. Nejprve vytvoří kategorie, poté získá hodnoty a klíče. Nakonec vytvoří otázky, které jsou zkopírovány do finálního adresáře.

3. Offline verze hry

Při implementaci online verze hry jsme narazili na jisté problémy. Naše online verze používá klient-server architekturu. Potřebujeme proto spolehlivou cloudovou platformu, která by umožňovala dlouhodobý běh serveru. Takovou platformu není úplně jednoduché nalézt. Z toho důvodu jsme se rozhodli implementovat i offline verzi hry. Tu jsme sice vytvořili až po online verzi, ale jelikož je její implementace jednodušší, podíváme se na ni dříve. Získané poznatky potom použijeme v další kapitole, kde se budeme věnovat online verzi hry.

3.1 Programovací jazyk

K vývoji aplikací pro Android lze použít více programovacích jazyků. Mezi nejpoužívanější patří Java, Kotlin a C++. Java je robustní a osvědčený jazyk s širokou podporou a mnoha knihovnami. V posledních letech však zaznamenává ve vývoji mobilních aplikací jistý útlum a své místo přepouští Kotlinu. Kotlin je moderní jazyk, oficiálně podporovaný Googlem, který snižuje opakování kódu a zlepšuje bezpečnost aplikací. V našem případě jsme vybrali Javu. Důvodem byla její rozsáhlá knihovnu a zejména osobní zkušenost s tímto programovacím jazykem.

3.2 Instance aplikace

V této sekci si ukážeme, z jakých komponent se vlastně taková mobilní aplikace skládá.

3.2.1 Activity

Základním prvkem uživatelského rozhraní v aplikacích pro Android je tzv. "Activity". Jedná se o obrazovku, která obsahuje uživatelské rozhraní pro interakci s uživatelem. Každá aplikace může mít jednu nebo více aktivit. Každá aktivita má svůj životní cyklus, který zahrnuje různé stavy, jako je vytvoření, spuštění, pozastavení, obnovení a zničení. Jednotlivé aktivity se skládají z fragmentů.

3.2.2 Fragmenty

Fragmenty jsou další komponentou v aplikacích pro Android. Jsou to modulární a znovupoužitelné části uživatelského rozhraní, které mohou být dynamicky přidávány nebo odebrány z aktivity za běhu aplikace. Fragmenty umožňují rozdělit uživatelské rozhraní na menší a nezávislé části, což usnadňuje správu a údržbu kódu. Každý fragment má také svůj vlastní životní cyklus, který je podobný životnímu cyklu aktivity. Zahrnuje stavy, jako je vytvoření, spuštění, pozastavení, obnovení a zničení.

3.2.3 Životní cykly

Následující diagram představuje životní cyklus aktivit a fragmentů a vztah mezi nimi. Pro nás je v tuto chvíli nejdůležitější, co se stane, je-li v důsledku systémové změny aktivita zničena a znovu vytvořena. Příkladem systémové změny je přepnutí světlého motivu zařízení na tmavý či rotace obrazovky. V takovém případě je u každého fragmentu spojeného s aktivitou a u samotné aktivity volána metoda **onSaveInstanceState(@NonNull Bundle outState)**. Do objektu outState si můžeme uložit data, která při novém vytvoření aktivity použijeme.

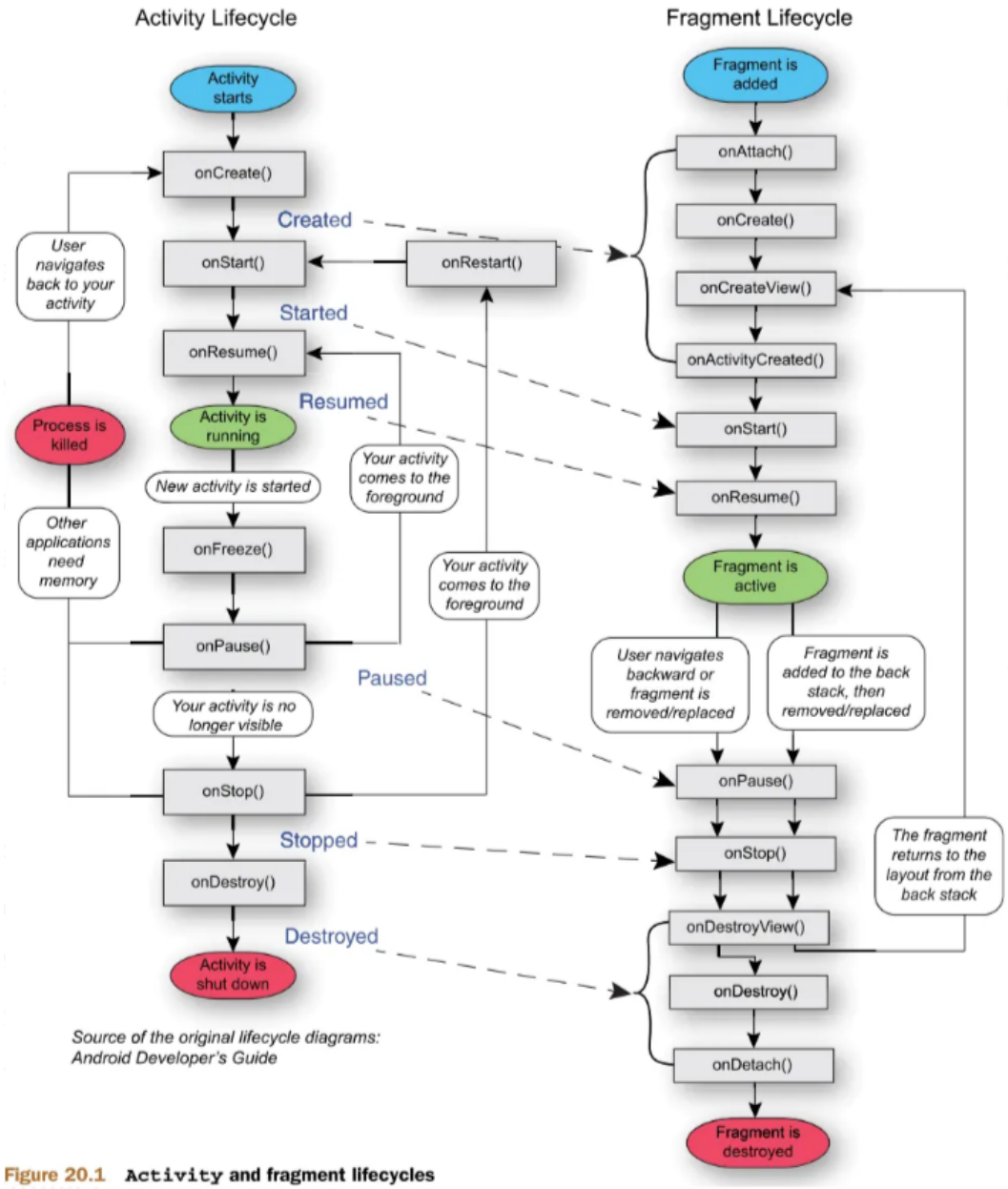


Figure 20.1 `Activity` and fragment lifecycles

Obrázek 3.1: Životní cyklus aktivity a fragmentu

3.3 Responsivní layout

Při vývoji mobilních aplikací je jedním z klíčových prvků responzivní layout. Umožňuje aplikaci přizpůsobit se různým velikostem displeje a orientacím zařízení, což je zásadní pro uživatelskou přívětivost a celkový dojem z aplikace.

3.3.1 Lineární layout

Lineární layout je jedním z nejjednodušších způsobů, jak vytvořit responzivní rozložení v Android aplikacích. Při použití lineárního layoutu jsou prvky uspořádány buď vodorovně nebo svisle za sebou. Tento přístup je snadný a intuitivní. Jeho hlavní nevýhodou je obtížnost řízení rozmístění prvků na obrazovce, zejména pokud se jedná o komplexní layouty. (4)

3.3.2 Relativní layout

Relativní layout je další možností pro tvorbu responzivního rozložení v Android aplikacích. Tento layout umožňuje definovat vztahy mezi jednotlivými prvky pomocí jejich relativní pozice vůči sobě nebo k okraji obrazovky. Relativní layout je flexibilnější než lineární layout, protože umožňuje jednodušeji řídit polohu prvků na obrazovce. (5)

3.3.3 Constraint layout

Constraint layout je relativně novým přístupem k tvorbě responzivního layoutu v Android aplikacích. Tento layout kombinuje flexibilitu relativního layoutu s výkonností a jednoduchostí lineárního layoutu. Hlavní výhodou Constraint layoutu je možnost definovat vztahy mezi prvky pomocí omezujících podmínek (constraints), což umožňuje vytvářet komplexní a flexibilní rozložení s minimálním úsilím. Díky omezujícím podmínkám můžeme jednoduše řídit, jak se prvky chovají při změně velikosti obrazovky nebo orientace zařízení. Navíc Constraint layout poskytuje nástroje pro náhled rozložení, což usnadňuje vizuální ladění layoutu a rychlé iterace během vývoje. (2)

3.3.4 Zvolený layout

Nejprvme jsme zkusili použít Relativní layout. Ten jsme hezky odladili na velikost obrazovky telefonu, na němž jsme aplikaci testovali. Bohužel se ukázalo, že takto odladěný layout je pro větší displeje nevhodný. Tento problém bychom mohli vyřešit tak, že pro větší displeje bychom definovali jiný Relativní layout. My jsme se však rozhodli použít Constraint layout, kde nám stačilo definovat jeden layout pro všechny velikosti displejů. Zároveň layout podporujeme světlý i tmavý režim zobrazení. Toho jsme docíli tak, že jsme si definovali pro každý režim zvláštní sadu barev, které se automaticky přepínají podle nastavení zařízení. Pro úplnost dodejme, že u malých displejů kvůli zachování hezkého designu nepodporujeme rotaci obrazovky. Následující obrázky ukazují vzhled naší aplikace na velkém a malém displeji.

Obrázek 3.2: Ukázka velké obrazovky v tmavém režimu

Obrázek 3.3: Ukázka malé obrazovky ve světlém režimu

3.4 StartingActivity

Nyní máme již dostatečné znalosti, abychom se podívali na první aktivitu v naší offline verzi hry. Jedná se o jednoduchou aktivitu, která obsahuje 2 fragmenty.

3.4.1 WelcomePage

Tento fragment reprezentuje úvodní obrazovku hry. Obsahuje 2 tlačítka.

Obrázek 3.4: Úvodní obrazovka

- *Hrát sám* - Po stisknutí se spustí nová aktivita. Tato aktivita se jmenuje BrowsingActivity a umožňuje uživateli procházet si vygenerované otázky
- *O hře* - Po stisknutí je uživateli zobrazen fragment FriendGamePage. Princip zobrazení je následující. Stisk tlačítka informuje aktivitu o tom, že má skrýt současný zobrazený fragment a zobrazit nový

3.4.2 FriendGamePage

Tento fragment je určený pro vytvoření hry. Uživatel zadá počet hráčů a jejich jména. Fragment používá zmíněnou metodu **OnSaveInstanceState**, aby si uložil index zvýrazněného počtu hráčů. Zadaná jména hráčů se ukládají automaticky.

Obrázek 3.5: Vytvoření hry

- *Zahájit hru* - Po stisknutí se ověří, že jsou zadaná jména validní. Pokud je nějaké jméno prázdné, nebo jsou-li 2 jména stejná, je uživateli zobrazen alertDialog s příslušnou chybovou hláškou. Jsou-li jména validní, spustí se nová aktivita s názvem GameActivity. Ještě dopňme, že informaci o tom, že je zobrazen alertDialog si při zničení aplikace také musíme uložit
- *Zpět* - Po stisknutí je uživateli přesunut zpět na fragment WelcomePage

3.5 GameActivity

Druhou aktivitou, na kterou se podíváme, je samotná hra. Oproti předchozí aktivitě nám přibyde nějaká logická vrstva. Musíme vybírat otázky, kontrolovat

správnost odpovědí či pamatovat si skóre hráčů. To ale není takový problém, větším oříškem se může zdát, jak vyřešit zničení a znovuvytvoření aktivity. U předchozí aktivity si stačilo zapamatovat index zvýrazněného počtu hráčů a informaci o tom, zda je zobrazen alertDialog. Tentokrát si však potřebujeme zapamatovat mnohem více informací. Zároveň při hře potřebujeme nějak šikovně informovat příslušné fragmenty o tom, že má být jejich obsah změněn. Například fragment zobrazující otázku musí po dokončení kola zobrazit novou otázku. Efektivním řešením těchto problémů je použít návrhový vzor Model–view–viewmodel. K jeho popisu si nejprve musíme představit 2 třídy.

3.5.1 ViewModel

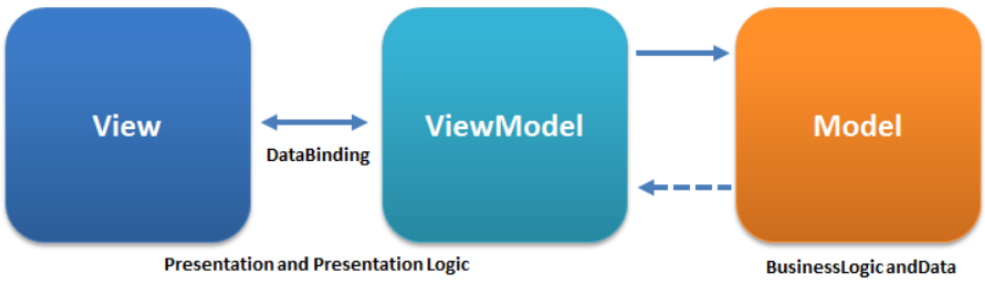
ViewModel slouží jako mezivrstva mezi aktivitou a logikou hry. Nejdůležitější vlastností ViewModelu je jeho nezávislost na životním cyklu aktivity. To znamená, že při zničení a znovuvytvoření aktivity je obsah ViewModelu zachován. Díky tomu můžeme ve Viewmodelu vytvořit instanci třídy, která slouží jako logika hry. (6)

3.5.2 MutableLiveData

MutableLiveData je specializovaná třída poskytovaná architekturou Android Jetpack, která umožňuje komunikaci mezi ViewModelem a uživatelským rozhraním pomocí datových proudů. Jedná se o pozorovatelné objekty, které mohou být modifikovány a upozorní své pozorovatele (např. aktivity nebo fragmenty) o změnách. (9)

3.5.3 Model–view–viewmodel

Model-view-viewModel (MVVM) funguje následujícím způsobem: View, což jsou v našem případě jednotlivé fragmenty, interagují s uživatelem a zaznamenávají jeho akce, jako je například odpověď na otázku. Tato odpověď je předána ViewModelu, který ji předá logice hry. Logika hry pak vyhodnotí odpověď a aktualizuje MutableLiveData ve ViewModelu. Je důležité poznamenat, že tato aktualizace probíhá asynchronně. Obsah fragmentů, které pozorují tyto aktualizované datové proudy, je následně upraven podle nových informací. (9)



Obrázek 3.6: Model–view–viewmodel

3.5.4 CountDownTimer

Poslední úpravou je přidání časového limitu pro zobrazení některých fragmentů. To je kvůli plynulejšímu hernímu zážitku, aby hráč nemusel u každého fragmentu klikat na tlačítko Další. K omezení doby zobrazení určitých fragmentů využíváme třídu CountDownTimer. Ta umožňuje spouštět časovač s definovaným časovým intervalem. Instanci třídy CountDownTimer vytvoříme přímo v GameActivity. Zbývající čas do změny fragmetu si uložíme do třídy ViewModel. Při každém kroku budeme tuto hodnotu aktualizovat. Tím zařídíme správnou inicializaci CountDownTimer při znovuvytvoření aktivity. Zároveň CountDownTimer odstraníme pokaždé, když se aplikace přesune do pozadí. Při opětovném přesunutí aplikace do popředí jej znovu vytvoříme.

3.5.5 Fragmenty

GameStart

Tento fragment s prázdným layoutem používám při zahájení hry. GameActivity při každé změně fragmentu schová aktuálně zobrazený fragment a zobrazí nový. Při startu hry schová GameActivity prázdný fragment. Doba zobrazení prázdného fragmentu je nastavena na 0 vteřin.

RoundStart

Fragment zobrazující číslo aktuálního kola. Je zobrazen po dobu 3 vteřin.

Obrázek 3.7: Číslo kola

RoundPlayer

Tento fragment zobrazuje jméno hráče, který je na tahu. Je také zobrazen po dobu 3 vteřin.

Obrázek 3.8: Hráč na tahu

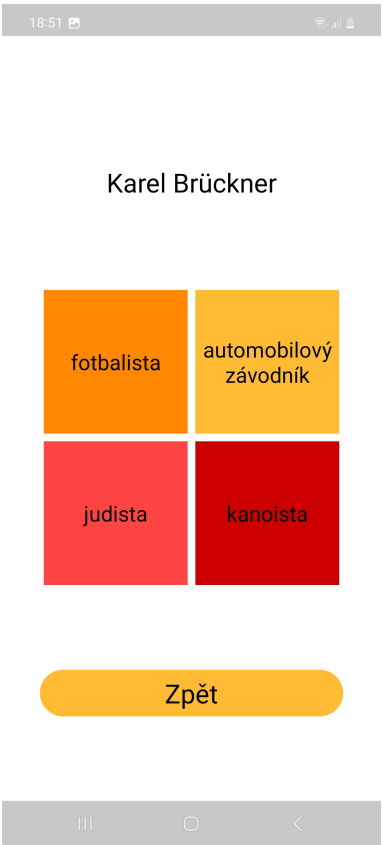
Question

Fragment zobrazující otázku. Každá otázka má 10 podotázek. Stisknutím otazníku u podotázky se hráči zobrazí detail podotázky včetně 4 možných odpovědí. Podotázky, na které již hráči odpovídali, obsahují místo otazníku správnou odpověď. Hráč může také využít možnosti pasovat stisknutím tlačítka pasovat. Hráč má neomezený čas na odpověď. Obrázek se nachází na další stránce.

QuestionDetail

Tento fragment zobrazuje hráči detail otázky. Hráč má na výběr ze 4 odpovědí. Stisknutím tlačítka zpět se vrátí zpět na fragment s otázkou. Obrázek se nachází na další stránce.

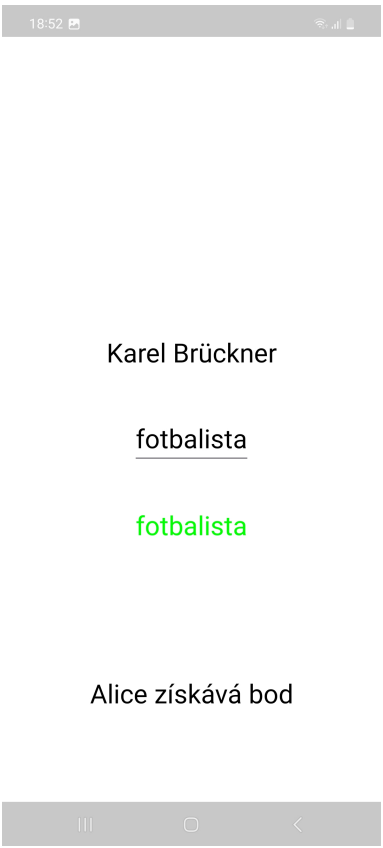
Obrázek 3.9: Otázka



Obrázek 3.10: Detail otázky

Answer

Tento fragment zobrazuje hráčovu odpověď na otázku. Je-li odpověď správná, má správná odpoveď zelenou barvu. V opačném případě má správná odpoveď barvu červenou. Pokud hráč pasuje, zobrazí se místo správné odpovědi informace o tom, že hráč pasuje. Tento fragment je zobrazen po dobu 3 vteřin.



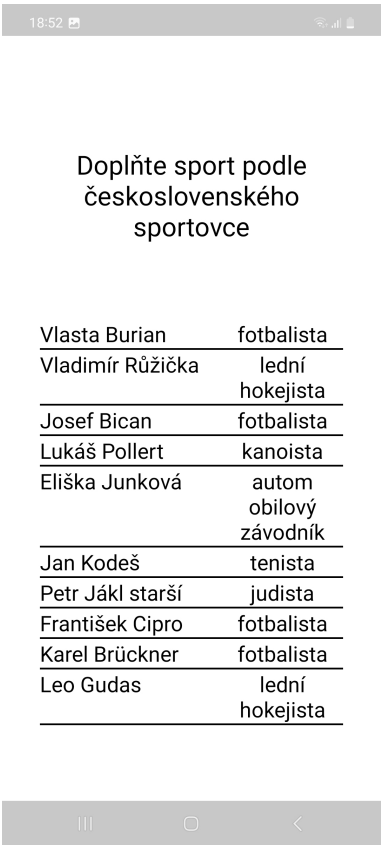
Obrázek 3.11: Hráč odpověděl

Question

Na konci kola je hráčům opět zobrazen fragment Question. Tentokrát již ale obsahuje správné odpovědi na všechny podotázky. Tento fragment je zobrazen po dobu 10 vteřin. Obrázek se nachází na další stránce.

Score

Tento fragment zobrazuje skóre hráčů. Pokud hra ještě neskončila, je zobrazen po dobu 5 vteřin, potom začne nové kolo. Hra končí ve chvíli, kdy jeden z hráčů získá alespoň 20 bodů. V takovém případě jsou k fragmentu přidána 2 tlačítka. První začne novou hru, druhé ukončí GameActivity a spustí StartingActivity. Obrázek se nachází na další stránce.



Obrázek 3.12: Řešení

Obrázek 3.13: Skóre

3.6 BrowsingActivity

Poslední aktivitou v offline verzi mobilní hry Desítka je BrowsingActivity. Tato aktivita umožňuje hráčům procházet si otázky ve hře. Stejně jako předchozí aktivita používá ViewModel pro uložení otázky. ViewModel také obsahuje instanci třídy, která má na starost výběr otázky. Pro zobrazení vyhodnocení uživatelovy odpovědi po dobu 3 vteřin používáme CountDownTimer.

3.6.1 Fragmenty

BrowsingStart

Jedná se o fragment s prázdným layoutem. Jeho účel je stejný jako v případě GameActivity. Při startu aktivity je tento prázdný fragment schován.

BrowsingQuestion

Tento fragment zobrazuje hráči vybranou podotázku z otázky se 4 možnostmi. Výchozí chování aktivity je takové, že hráči je postupně zobrazeno všech 10 podotázek vybrané otázky. Poté je vybrána nová otázka. Stisknutím tlačítka Jiná otázka je pro hráče nová otázka vybrána okamžitě. Stisknutím tlačítka Ukončit dojde k ukončení aktivity a spuštění StartingActivity.

Obrázek 3.14: Otázka

BrowsingAnswer

Tento fragment vyhodnotí správnost uživatelovy odpovědi. Je téměř totožný s fragmentem Answer z GameActivity. Pouze nikdy nezobrazí informaci o tom, že hráč pasuje.

3.7 Uživatelská dokumentace

V adresáři DesitkaOffline je soubor desitka_offline.apk. Jedná se již o vytvořenou aplikaci, kterou by mělo jít nainstalovat na zařízení. Vytvořil jsem 2 videa, která instalaci popisují. První **video** ukazuje, jak aplikaci nainstalovat na mobilní zařízení. Druhé**video** ukazuje, jak použít BlueStacksX a nainstalovat aplikaci na počítač či notebook. BlueStacksX je platforma, která umožňuje uživatelům spouštět mobilní aplikace a hry na počítačích s Windows, či macOS.

3.8 Programátorská dokumentace

Samotný program se skládá z několika tříd v několika balíčcích. Podrobná dokumentace je v adresáři javadoc na následujícím **odkazu**.

3.8.1 Balíčky

desitka.activities

Tento balíček obsahuje aktivity použité v aplikaci. Jedná se o BrowsingActivity, GameActivity a StartingActivity.

desitka.fragments.browsing

Balíček pro fragmenty použité v BrowsingActivity.

desitka.fragments.game

Balíček s fragmenty použitými v GameActivity.

desitka.fragments.starting

Balíček pro fragmenty použité ve StartingActivity.

desitka.interfaces

Tato sekce obsahuje interfacy použité v aplikaci. Jednotlivé interfacy obsahují enumy. Každý enum představuje stav, ve kterém se aktivita nachází. Pro každý stav je určen příslušný fragment.

desitka.logic.browsing

Tento balíček obsahuje jednoduchou logiku pro BrowsingActivity. Balíček obsahuje také BrowsingViewModel.

desitka.logic.game

Tento balíček obsahuje logiku pro GameActivity. Balíček obsahuje třídu GameController, která pro každé kolo vybere otázku a vytvoří instanci třídy Round. Tato instance má na starost logiku jednoho kola. Jednotliví hráči jsou reprezentováni instancí třídy Player. Balíček také obsahuje GameViewModel použitý v GameActivity.

desitka.logic.game.liveData

Tento balíček obsahuje MutableLiveData použitá ve GameViewModelu pro GameActivity. Dodejme, že GameViewModel používá i některé další třídy jako MutableLiveData. Například třída Question pro reprezentaci otázky je také použita jako MutableLiveData

desitka

V tomto balíčku najdeme třídu reprezentující otázku použitou ve hře. Jedná se o stejnou třídu, kterou jsme použili při generování otázek. Dále zde najdeme třídu pro načtení otázky z Json souboru. Tato třída využívá knihovnu Gson. Poslední třídou v balíčku je třída pro uložení uživatelových preferencí. Jediným úkolem této třídy je uložit si počet hráčů, který uživatel zadal při vytváření hry a v okamžiku, kdy je uživateli znovu zobrazen fragment sloužící k vytvoření hry, nastavit počet hráčů na uloženou hodnotu. (3)

3.8.2 MutableLiveData

GameActivity

- **Start**
  - Tato třída obsahuje číslo kola a objekt Turn.
- **Turn**
  - Tato třída obsahuje jméno hráče, který je na tahu a jeho skóre. Dále třída obsahuje počet hráčů v daném kole a počet hráčů v celé hře.
- **Answer**
  - Tato třída obsahuje index podotázky, jméno odpovídajícího hráče, index hráčovy odpovědi, index správné odpovědi a objekt Turn. V případě, že hráč pasuje, je index podotázky nastaven na -1.
- **Question**
  - Tato třída reprezentuje otázku ve hře.
  - Na začátku kola si nastavíme hodnoty indexů všech správných odpovědí na -1. Po každé odpovědi hráče, pokud hráč nepasuje, si potom pro příslušnou podotázku zaznamenáme hodnotu správného indexu odpovědi.

- **Evaluation**
  - Tato třída obsahuje skóre hráčů a informaci o tom, zda-li je konec hry.
- **Integer - questionDetailID**
  - Index podotázky, jejíž detail si hráč zobrazil.
- **GameState - gameFragmentChange**
  - Stav, do kterého má hra přejít.
- **GameState - displayedGameFragment**
  - Současný stav, ve kterém se hra nachází. Tento objekt využíváme pro nastavení CountDownTimer. Jakmile CountDownTimer skončí, tak nastaví hodnotu gameFragmentChange. GameActivity poté přejde do nového stavu a zobrazí příslušný fragment. Potom GameActivity aktualizuje hodnotu displayedGameFragment a CountDownTimer začne nový odpočet.

BrowsingActivity

- **Question.SubQuestion**
  - Vybraná podotázka z otázky.
- **String - questionText**
  - Text otázky, který je společný pro všechny podotázky.
- **Integer - playerAnswer**
  - Index hráčovy odpovědi.
- **BrowsingState - BrowsingFragmentChange**
  - Stav, do kterého má hra přejít.
- **BrowsingState - displayedBrowsingFragment**
  - Současný stav, ve kterém se hra nachází.

4. Online verze hry

V této kapitole si ukážeme, jak předělat naši offline verzi hry na online verzi. Nejprve si ukažme, v čem se online verze nejvíce liší.

4.1 Rozdíly oproti offline verzi hry

4.1.1 Server

Nejpatrnějším rozdílem je, že tentokrát nehrají hráči hru na jednom zařízení, ale každý z nich má své vlastní. Musíme tedy nějak propojit jednotlivá mobilní zařízení (klienty) mezi sebou. K tomu použijeme klient-server architekturu. Veškerou logiku hry přesuneme na server. Server bude mít na starost výběr otázky, validaci odpovědí či počítání skóre. Klienti budou mít na starost pouze zpracování hráčovy akce, například odpovědi na otázku a posláni ji serveru. Server potom odpověď vyhodnotí a pošle ji zpět klientovi. Klient prostřednictvím grafického uživatelského rozhraní zobrazí hráči odpověď serveru.

4.1.2 Java Socket

Pro komunikaci mezi klienty a serverem prostřednictvím internetu jsme si vybrali knihovnu Java Socket. Tato knihovna umožňuje komunikaci mezi aplikacemi pomocí soketů, což jsou programová rozhraní pro síťové spojení. Do naší aplikace jsme museli přidat nové vlákno, které má na starost komunikaci. Pro výměnu dat jsme zvolili formát Json.(1)

4.1.3 Formát Json

Pro sjednocení zpráv, které si server a klient mezi sebou vyměňují jsme vytvořili několik tříd. Pro serializaci používáme knihovnu Gson.

client

Toto jsou zprávy, které posílá klient serveru.

- **MyAnswer**
  - Hráčova odpověď na otázku. Obsahuje index podotázky a index hráčovy odpovědi.
- **MyJoining**
  - Hráčův požadavek na připojení se do hry. Obsahuje jméno hráče, kód hry a počet hráčů ve hře. Prvním atributem MyJoiningu je RequestType. Jedná se o enum, který může mít 3 hodnoty:

        * JOIN_ONLINE_GAME
            · Připojení se do online hry, která je pro 2 hráče.
        * JOIN_FRIEND_GAME
            · Připojení se do hry s přáteli na základě kódu hry.
        * CREATE_FRIEND_GAME
            · Vytvoření nové přátelské hry pro zadaný počet hráčů.

server

Toto jsou zprávy, které posílá server klientovi.

- **Answer**
  - Index podotázky, jméno hráče, index hráčovy odpovědi, index správné odpovědi a objekt Turn.
- **Evaluation**
  - Vyřešená otázka, skóre hráčů a informace o tom, jestli hra skončila.
- **Joining**
  - Kód hry, počet hráčů a čas vytvoření hry. Prvním atributem Joiningu je JoiningResult. Jedná se o enum, který může mít 3 hodnoty:
    * JOINED
        · Úspěšné připojení do hry.
    * GAME_NOT_FOUND
        · Pro zadaný kód hry nebyla hra nalezena.
    * NAME_ALREADY_JOINED
        · Jiný hráč ve hře má stejné jméno.
- **Start**
  - Číslo kola, nevyřešená otázka (indexy správných odpovědí pro každou podotázkou jsou nastaveny na -1), objekt Turn.
- **Waiting**
  - Informace o tom, jestli se do hry již připojili všichni hráči a seznam hráčů.
- **Turn**
  - Jméno hráče na tahu, počet hráčů v kole, počet hráčů ve hře, informace o tom, jestli je hráč na tahu a skóre hráče. Poslední 2 hodnoty se pro každého hráče určují individuálně.

4.1.4 Neaktivita hráče

Dalším rozdílem oproti offline verzi hry je, že tentokrát nemůžeme dát hráčům neomezený čas na odpověď. Uvažme, že hráč, který je právě na tahu aplikaci ukončí. Server by pořád čekal na odpověď, která by nikdy nepřišla. Z toho důvodu jsme se rozhodli dát každému hráči 30 vteřin na odpověď. Pokud hráč do 30 vteřin nevykoná žádnou akci, pošle klient serveru informaci o tom, že hráč pasuje bez žádné aktivity. Jako index podotázky v Json objektu Answer se použije hodnota -10. Pokud server třikrát za sebou obdrží od klienta zprávu o tom, že hráč pasuje bez projevení aktivity, je hráč vyřazen ze hry. Pokud server neobdrží od klienta žádnou zprávu do 31,5 vteřiny (30 vteřin na odpověď + EXTRA_TIME), je hráč vyřazen ze hry.

4.1.5 GameTimerThread

V offline verzi hry nám vyhovovalo zastavit Countdowntimer pokaždé, když se aplikace přesunula do pozadí. Tentokrát jsme v jiné situaci. Potřebujeme zajistit, že Coundowntimer běží, i když je aplikace na pozadí. K tomuto účelu jsme vytvořili ještě jedno vlákno GameTimerThread. Toto vlákno obsahuje Countdowntimer, jenž má po skončení časového limitu za úkol aktualizovat stav, ve kterém se hra nachází. A to i když je aplikace na pozadí.

4.1.6 Oprávnění

Posledním rozdílem oproti offline verzi hry je, že online verze vyžaduje více oprávnění. Konkrétně vyžaduje oprávnění pro přístup k internetu. To je z toho důvodu, aby aplikace mohla komunikovat se serverem. Druhým vyžadovaným oprávněním je potom monitorování připojení k internetu. Jakmile není internet k dispozici, tak aplikace vytvoří alertDialog s chybovou hláškou, že aplikace nemůže bez internetového připojení správně fungovat.

4.2 StartingActivity

První ze dvou aktivit v online verzi hry je StartingActivity. Ta se skládá z následujících fragmentů.

4.2.1 WelcomePage

Tento fragment reprezentuje úvodní obrazovku hry. Obsahuje 3 tlačítka.

- *Hrát sám* - Po stisknutí se spustí GameActivity. Na server je poslán požadavek na připojení se do online hry.
- *Hrát s přáteli* - Po stisknutí aktivita schová aktuální fragment a zobrazí FriendGamePage fragment.
- *Nastavení* - Po stisknutí aktivita schová aktuální fragment a zobrazí fragment Settings. V něm si uživatel může změnit jméno a také nastavit adresu a port serveru.

Obrázek 4.1: Úvodní obrazovka

4.2.2 FriendGamePage

Tento fragment slouží k nastavení hry s přáteli. Obrázek fragmentu se nachází na další stránce.

- *Vytvořit hru* - Nahradí aktuální fragment za fragment CreateGame
- *Připojit se* - Nahradí aktuální fragment za fragment JoinGame
- *Zpět* - Nahradí aktuální fragment za fragment WelcomePage.

4.2.3 FriendGameCreate

Tento fragment slouží k vytvoření přátelské hry. Obrázek se nachází na další stránce.

- *Vytvořit hru* - Po stisknutí tlačítka je spuštěna GameActivity a na server je poslán požadavek na vytvoření nové přátelské hry. Je-li požadavek úspěšný, je uživateli zobrazen GameStart fragment z GameActivity. Na něm hráč vidí počet připojených hráčů a také kód hry, který může sdílet se svými přáteli.
- *Zpět* - Nahradí aktuální fragment za fragment FriendGamePage.

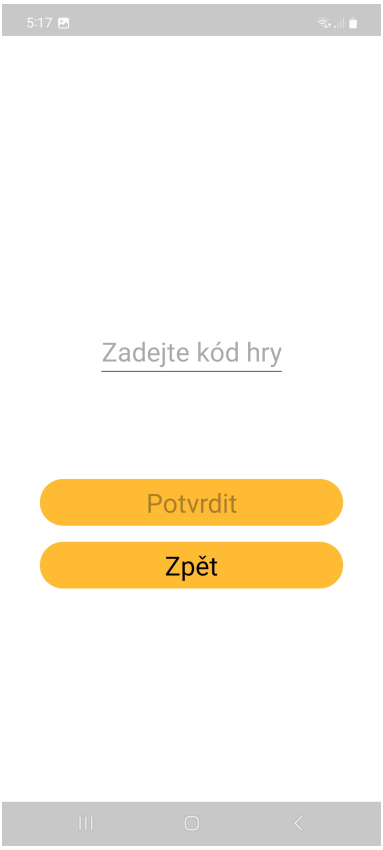
Obrázek 4.2: Hra s přáteli

Obrázek 4.3: Vytvořit hru

4.2.4 JoinGame

Tento fragment slouží k připojení se do přátelské hry. Vyžaduje zadání osmimístného kódu hry.

- *Potvrdit* - Toto tlačítko je možné stisknout po zadání osmimístného kódu hry. Po stisknutí je spuštěna GameActivity a na server je poslán požadavek na připojení se do hry na základe zadaného kódu.
- *Zpět* - Nahradí aktuální fragment za fragment FriendGamePage.



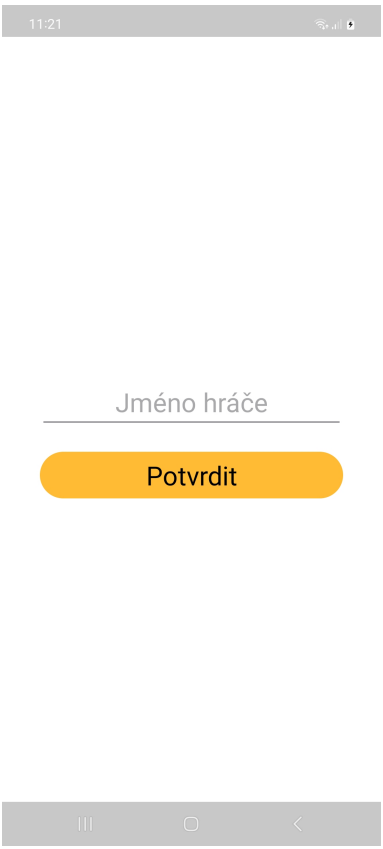
Obrázek 4.4: Připojit se do hry

4.2.5 SetName

Tento fragment slouží k zadání jména hráče. je zobrazen při prvním spuštění aplikace. Obrázek fragmentu se nachází na další stránce.

4.2.6 Settings

Tento fragment slouží k upravení jména hráče. Také umožňuje změnu nastavení adresy a portu serveru. Obrázek je na další stránce.



Obrázek 4.5: Připojit se do hry

Obrázek 4.6: Nastavení

4.3 GameActivity

Tato aktivita obsahuje podobné fragmenty jako stejnojmenná aktivita v offline verzi hry. Abychom nepřidávali obrázky stejných fragmentů znovu, přidáme jenom obrázky takových, které offline verze hry nemá.

4.3.1 ServerWaiting

Jedná se o fragment s prázdným layoutem, který používáme, když čekáme na odpověď od serveru. Na začátku hry jsme ve stavu GAME_JOINING a čekáme na odpověď od serveru. Pokud do 3 (2*EXTRA_TIME) vteřin neobdržíme od serveru nějakou odpověď, vytvoříme alertDialog s informací, že nastala chyba při komunikaci se serverem.

Tento layout používáme ještě ve stavu SHOW_ANSWER_WAITING. To je stav, do kterého jsme přešli, pokud jsme do 30 vteřin nedostali hráčovu odpoveď. Opět počkáme 2*EXTRA_TIME a v případě, že nedostaneme od serveru žádnou odpověď, vytvoříme alertDialog s informací o chybě při komunikaci. První EXTRA_TIME představuje časový limit, který server čeká na odpověď od klienta, pokud hráč nevykoná žádnou akci. Druhý EXTRA_TIME je čas, ve kterém musí server informovat všechny klienty o akci hráče, který je na tahu. To zahrnuje i případ, kdy se hráč odpojí.

4.3.2 GameStart

Tento layout používají hned 2 stavy hry. Pro WAITING_FOR_PLAYERS slouží layout jako čekací místnost, kde se postupně shromažďují hráči. Na připojení se do hry mají hráči od okamžiku založení hry 60 vteřin. Pokud se do 60 vteřin hra nenaplní, je zrušena. Hráčům je zobrazen AlertDialog s chybovou hláškou, že hráči nebyli nalezeni. Jakmile je hra naplněna, přejde aktivita do stavu GAME_START. V tomto stavu se na fragmentu zobrazí v dolní části informace o tom, že hra začíná. Obrázek je na stránce níže.

4.3.3 RoundStart

Jedná se o stejný fragment jako v offline verzi hry. Zobrazuje se vždy na začátku kola po dobu 3 vteřin a je nahrazen fragmentem RoundPlayer.

4.3.4 RoundPlayer

Jedná se o stejný fragment jako v offline verzi hry. Je zobrazen po dobu 3 vteřin a potom je nahrazen fragmentem Question.

4.3.5 Question

Jedná se také o stejný fragment jako v offline verzi hry. Jediný rozdíl je v tom, že odpovídat může pouze hráč, který je právě na tahu. U ostatních hráčů nemají stisknutí tlačítka Pasovat a otazníků u podotázek žádný efekt. Hráč má na odpověď 30 vteřin.

Obrázek 4.7: Hra začíná

4.3.6 QuestionDetail

Tento fragment je totožný s fragmentem v offline verzi hry. Stejně jako v případě fragmentu Question je tento fragment zobrazen po dobu 30 vteřin. Po uplynutí časového limitu pošle klient serveru zprávu o tom, že hráč pasuje bez jakékoliv aktivity. To platí bez ohledu na to, jestli je odpovídajícímu hráči při vypršení času na odpověď zobrazen fragment Question nebo fragment QuestionDetail.

4.3.7 Answer

Jedná se opět téměř o totožný fragment s offline verzí. Jedinou změnou je, že zobrazujeme i informaci o tom, že se hráč odpojil. Pokud se hráč odpojil, tak má index hráčovy odpovědi hodnotu -100. Fragment s odpovědí je zobrazen po dobu 3 vteřin.

4.3.8 Score

Tento fragment je opět totožný s fragmentem v offline verzi hry. Skóre hráčů je zobrazeno po dobu 5 vteřin. Jakmile hra skončí může hráč ukončit aktivitu stisknutím tlačítka Ukončit. Stisknutím tlačítka Hrát znovu se může hráč připojit do nové hry. Pokud hrál online verzi hry, je přidán do libovolné hry. Hrál-li hráč hru s přáteli, je přidán do stejné hry. Pro start této hry se tedy musí připojit všichni původní hráči. Popřípadě nějaký nový hráč, který ale zná kód hry.

4.4 Uživatelská dokumentace

4.4.1 Spuštění aplikace

Pro hraní hry je potřeba udělat 3 základní kroky. Na následujícím **odkazu** je video s ukázkou postupu.

Server

Spuštění serveru je asi tím nejjednoduším krokem. Stačí mít nainstalovanou javu verze alespoň 8 a v adresáři Server potom zadat následující příkaz. Server se spustí na localhostu a naslouchá na portu 4444.

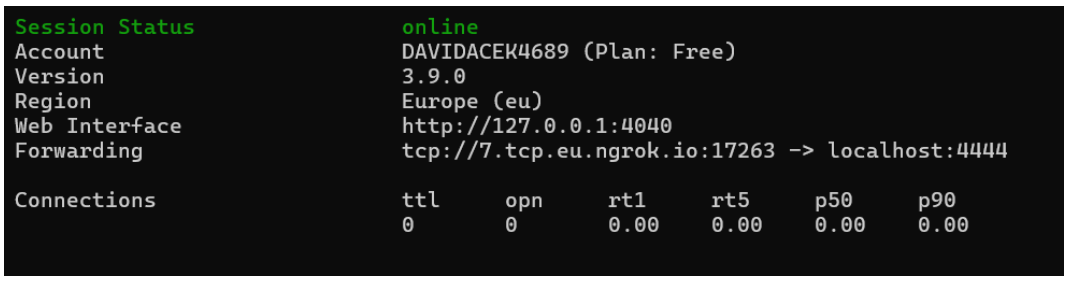
```
java -jar server-jar-with-dependencies.jar
```

Klient

Pro spuštění aplikace je potřeba stáhnout si apk soubor z repozitáře a nainstalovat jej.

Ngrok

Nakonec musíme vytvořit veřejný tunel ke spuštěnému serveru. Tento veřejný tunel použije mobilní aplikace pro komunikaci se serverem. K tomuto účelu použijeme nástroj ngrok. Nejprve si musíme vytvořit účet na **ngrok**. Účet je zcela zdarma. Nástroj ngrok nemusíme ani stahovat, neboť je přítomen v **repozitáři** práce. Nejprve musíme zadat náš autentizační token, který je vytvořen pro každého úspěšně registovaného uživatele. Potom stačí v terminálu zadat příkaz **ngrok tcp 4444**. Po zadání se zobrazí informace podobné obrázku níže.



Obrázek 4.8: Použití nástroje ngrok

Vidíme, že localhost na portu 4444 je namapovaný na následující adresu **0.tcp.eu.ngrok.io:16497**. Nyn musíme tyto informace předat naší aplikaci. K tomu stačí na úvodní stránce aplikace stisknout tlačítko Nastavení. Tím se nám zobrazí fragment Settings, kde můžeme jednoduše nastavit adresu a port.(7)

4.4.2 Nasazení aplikace

Serverovou část aplikace jsem se nakonec rozhodl nikam nenasazovat. Nepovedlo se mi najít spolehlivou bezplatnou cloudovou platformu, která by umožňovala dlouhodobý běh serveru. Vedoucí mé práce, pan doktor Mareček, mi nabídl nasazení serverové části na fakultní servery. Ovšem třeba pouze na půl roku. Takto omezená doba běhu serveru znemožňuje plné využití aplikace v dlouhodobém horizontu. Tím je také znemožněna distribuce aplikace prostřednictvím oficiálních kanálů jako Google Play Store.

Zároveň se domnívám, že výše uvedený postup spuštění aplikace není tak složitý. A dokonce má i své výhody. Například díky tomu, že je server responsibilní, může uživatel jednoduše kontrolovat komunikaci mezi serverem a klienty.

4.5 Programátorská dokumentace

4.5.1 JSON

JSON balíček je totožný pro server i pro aplikaci. Podrobná dokumentace je v adresáři javadoc na následujícím **odkazu**.

desitka.JSON

Tento balíček obsahuje třídu GsonParser, která slouží k převodu tříd do Json formátu a opačně. Balíček také obsahuje časový limit pro stavy, v nichž se hra nachází.

desitka.JSON.client

Tento balíček obsahuje JSON třídy, jejichž instance posílá klient serveru.

desitka.JSON.server

Tento balíček obsahuje JSON třídy, jejichž instance posílá server klientovi.

desitka.JSON.server.helper

Tento balíček obsahuje pomocnou třídu Turn, kterou používá server. Tato třída obsahuje v proměnné playerOnMove jméno hráče, který je na tahu. Pokud kolo skončilo, je v proměnné hodnota null.

4.5.2 Server

Podrobná dokumentace je v adresáři javadoc na následujícím **odkazu**.

desitka.server.questions

Tento balíček obsahuje jedinou třídu, která slouží k načtení otázek uložených v Json formátu.

desitka.server

Tato instance obsahuje třídu s main metodou pro spuštění server. Server běží na localhostu na portu 4444. Druhou třídou v tomto balíčku je GameManager, který má na starost správu jednotlivých her.

desitka.server.communication

Tento balíček má na starost komunikaci s klienty.

desitka.server.gameLogic

Tento balíček má na starost logiku hry. Podobně jako u offline verze hry máme třídu Game, která pro každé kolo vytvoří instanci tříd Round. Jednotliví hráči jsou opět reprezentováni instancí třídy Player. Každá instance třídy Player má vlastní CommunicationService, jenž umožňuje posílání zpráv klientovi.

4.5.3 Client

Podrobná dokumentace je v adresáři javadoc na následujícím **odkazu**.

client.activities

Tento balíček obsahuje třídy pro aktivity.

client.communication

Tento balíček obsahuje třídy sloužící ke komunikaci se serverem.

client.fragments.game

Tento balíček obsahuje fragmenty použití v GameActivity.

client.fragments.starting

Tento balíček obsahuje fragmenty použití ve StartingActivity.

client.gameLogic

Tento balíček slouží jako logika hry na straně klienta. Obsahuje GameViewModel. Dále obsahuje třídu GameModel, která vyytváří instance CommunicationService pro komunikaci se serverem a GameTimerThread pro správu CountdownTimeru.

client.sharedPreferences

Tento balíček obsahuje uložené preference hráče. Jedná se o jméno hráče, adresu serveru a port serveru.

4.5.4 MutableLiveData

JSON

Joining, **Waiting**, **Start**, **Turn**, **Answer**, **Evaluation**

Zbývající

- **Question**
  - Aktuální zobrazená otázka
- **GameState - gameStateChange**
  - Stav, do kterého má hra přejít
- **String - timerText**
  - Text CountDownTimeru
- **Integer - questionDetailID**
  - Index zvolené podotázky
- **Integer - connectionError**
  - Kód chyby spojení.

Závěr

V rámci této bakalářské práce jsme úspěšně implementovali mobilní verzi společenské vědomostní hry Desítka pro zařízení s operačním systémem Android verze 10 a výše. Vytvořili jsme dokonce 2 aplikace. První jde hrát bez internetového připojení. Druhá je určená pro online hraní a splňuje požadavky stanovené při zadání práce.

Druhým cílem práce bylo vytvořit otázky, které budou ve hře použity. V tomto ohledu jsme dle mého názoru také neselhali. Vytvořili jsme více adresářů s otázkami na základě minimální hranice průměrného měsíčního počtu zobrazení článků. Nakonec jsme vybrali adresář s hranicí 1024. Ten obsahuje celkem 314 otázek, z nichž 105 je unikátních. Otázky sice nepokrývají tak široké spektrum jako otázky z originální hry. Na druhou stranu mají ale dobře formulovaný text otázky, nabízené odpovědi spolu logicky souvisí a podařilo se nám zachovat přiměřenou obtížnost otázky.

Možná rozšíření práce

Offline verze mobilní aplikace hry se mi opravdu líbí. Možná by stálo za to kontaktovat společnost Mindok, jestli použití obrázku desítky ve hře a jména Desítka nějak neporušuje autorská práva. Pokud by bylo vše v pořádku, mohla by být práce umístěna na Google Play Store.

Generování otázek by šlo rozšířit tak, abychom vytvářeli i jiné typy otázek, které se vyskytují v originální deskové hře desítka. Konkrétně bychom mohli zkusit vytvořit řadící otázky. Úkolem takových otázek je uspořádání prvků či událostí v určitém pořadí. Dalším možným vylepšením by mohlo být rozdělit vygenerované otázky do různých okruhů a dát hráčům při hraní možnost výběru okruhu.

Seznam použité literatury

[1] BAELDUNG (2023). Baeldung.com. URL https://www.baeldung.com/a-guide-to-java-sockets.

[2] GOOGLE (2023). Android developers: Constraintlayout. URL https://developer.android.com/training/constraint-layout.

[3] GOOGLE (2023). Github: Google gson. URL https://github.com/google/gson.

[4] GOOGLE (2023). Android developers: Linearlayout. URL https://developer.android.com/guide/topics/ui/layout/linear.

[5] GOOGLE (2023). Android developers: Relativelayout. URL https://developer.android.com/reference/android/widget/RelativeLayout.

[6] GOOGLE (2023). Android developers: Viewmodel. URL https://developer.android.com/topic/libraries/architecture/viewmodel.

[7] JARUS, N. (2023). ngrok.com. URL https://ngrok.com/docs/secure-tunnels/.

[8] MEDIAWIKI (2023). mediawiki.org. URL https://www.mediawiki.org/wiki/API:Main_page.

[9] MICROSOFT (2023). Microsoft docs: Model-view-viewmodel (mvvm) design pattern. URL https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

[10] MINDOK (2018). mindok.cz. URL https://mindok.cz/hra/desitka/.

[11] MINDOK (2018). mindok.cz. URL https://www.mindok.cz/userfiles/files/pravidla/desitka_pravidla_web.pdf.

Seznam obrázků

A. Přílohy

A.1 Obsah elektronické přílohy

- Adresář DesitkaOffline
  - Zdrojový kód
  - Dokumentace Javadoc
- Adresář DesitkaOnline
  - Zdrojový kód
  - Dokumentace Javadoc
- Adresář QuestionGenerator
  - Zdrojový kód
  - Dokumentace Javadoc
- Adresář Server
  - Zdrojový kód
  - Dokumentace Javadoc